PROJECT DOCUMENTATION

DermalScan: AI Facial Skin Aging Detection

A report submitted for the project component of

Infosys Springboard Internship 6.0

Submitted By: S. Shanmukh Sai Nath

Mentor: Praveen

Project Documentation: DermalScan - AI Facial Skin Aging Detection

Problem Statement

The primary objective is to create a deep learning-based system capable of detecting and classifying various signs of facial aging. The system will identify conditions such as wrinkles, dark spots, and puffy eyes, as well as clear skin, by utilizing a pretrained EfficientNetB0 model. The project pipeline involves several stages: initial face detection using Haar Cascades, followed by custom image preprocessing and data augmentation techniques. Finally, the system will classify the identified signs and provide percentage predictions for each. A user-friendly, web-based frontend will be developed to allow users to upload facial images and view the results, which will be displayed with annotated bounding boxes and labels for clarity.

Expected Outcomes

Upon completion, the project is expected to deliver the following outcomes:

- **Detection and Localization:** The system will be able to detect and precisely locate facial features that indicate aging.
- **Classification:** It will classify the detected features into specific categories: wrinkles, dark spots, puffy eyes, and clear skin, using a trained Convolutional Neural Network (CNN) model.
- **Model Performance:** A robust EfficientNetB0 model will be trained and evaluated to ensure high classification accuracy.
- **Web Interface:** A functional web-based frontend will be available for users to upload images and view the annotated results.
- **Integrated Pipeline:** A complete backend pipeline will be integrated to process user images and return the annotated results seamlessly.
- **Data Export:** The system will allow users to export the annotated images and logs for further analysis or documentation.

Modules to be Implemented :-

The project will be developed across the following key modules:

- 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 Preparation and Preprocessing (Weeks 1-2)

This milestone focuses on building a high-quality dataset, which is the foundation for the deep learning model.

Module 1: Dataset Setup and Image Labeling

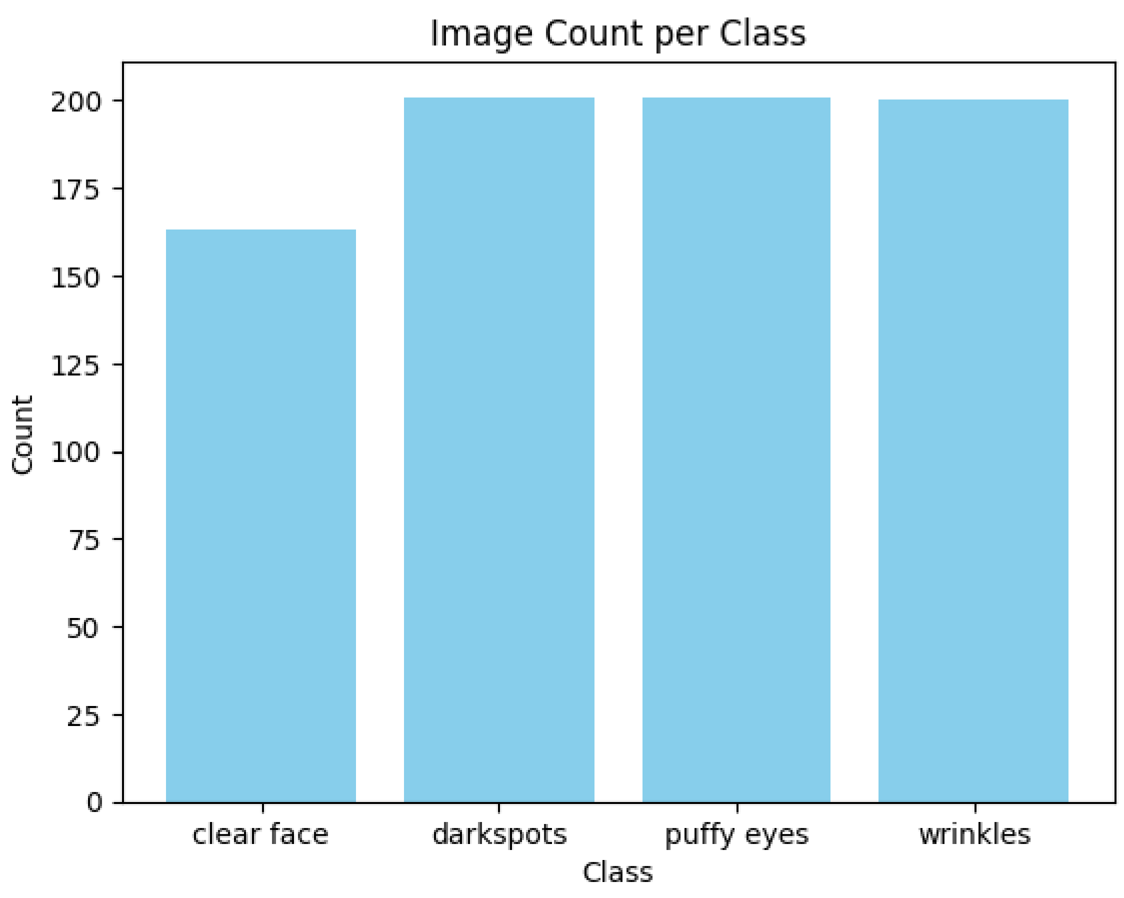
- **Tasks:**
  - Set up a dataset of facial images, either by creating a new one or using a preexisting collection.
  - Label the images with the appropriate classes:

wrinkles, **dark spots**, **puffy eyes**, and **clear skin**.

  - Ensure the dataset is clean and that the classes are balanced to avoid model bias.
- **Deliverables:**
  - A cleaned and accurately labeled dataset.
  - A class distribution plot to visualize the balance of data.
- **Evaluation:**
  - The dataset must have proper class balance.

    - Labels must be accurate and thoroughly inspected for quality.

Output:-

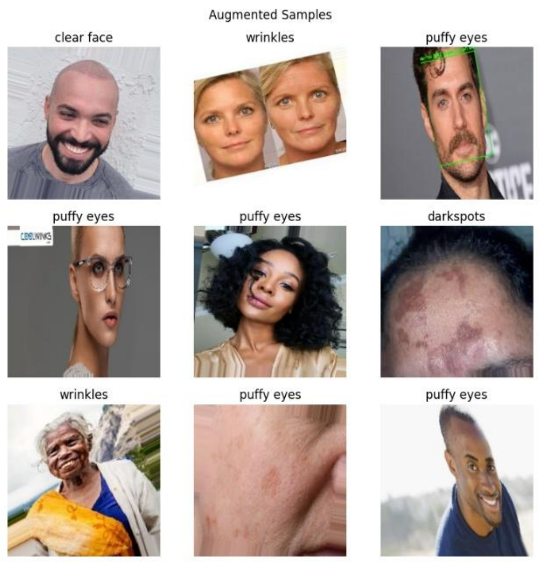


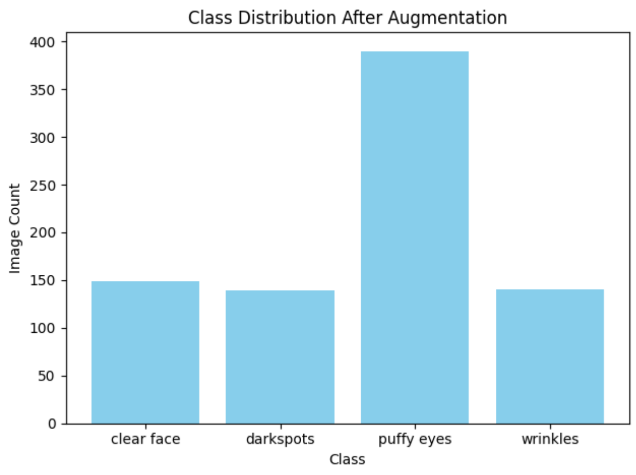
Module 2: Image Preprocessing and Augmentation

- **Tasks:**
    - Standardize all images by resizing them to 224x224 pixels and normalizing their pixel values.
    - Apply data augmentation techniques such as **flipping**, **rotation**, and **zooming** to artificially expand the dataset.
    - Convert the categorical class labels into a numerical format using one-hot encoding.

- **Deliverables:**
  - The final preprocessed and augmented dataset ready for training.
  - An augmentation script that includes visualizations of the transformed images.
- **Evaluation:**
  - The quality of augmentation and overall readiness of the dataset will be assessed.
  - The process must retain class diversity after augmentation is applied.

Output:-





Learning Reflections for Milestone 1

Completing this milestone provides foundational knowledge in machine learning project development.

- **Module 1 Reflections:** This module emphasizes the principle of **"garbage in, garbage out."** You learn that the performance of any AI model is heavily dependent on the quality of the training data. Key takeaways include the critical importance of creating a **balanced dataset** to prevent the model from becoming biased towards a specific class and the meticulous effort required for accurate image labeling. It provides hands-on experience in data curation and quality control.
- **Module 2 Reflections:** This module provides practical skills in image data manipulation. You learn that **preprocessing**, such as resizing and normalization, is a mandatory step to

standardize data for a neural network. The core lesson is understanding **data augmentation** as a powerful technique to combat overfitting, especially with smaller datasets. By artificially increasing the diversity of the training data, you learn how to help the model generalize better to new, unseen images. Finally, implementing **one-hot encoding** solidifies the understanding of how to prepare categorical data for classification tasks.

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

Module 3: Model Training with ResNet50
Tasks:

- Use a pretrained **ResNet50** model for transfer learning.
- Train using **categorical cross-entropy loss** and the **Adam optimizer**.
- Validate the model and visualize performance through **accuracy and loss curves**.

Deliverables:

- A trained **CNN model (.h5 file)**
- Plots showing **accuracy and loss trends** during training

Evaluation:

- Achieve at least **90% classification accuracy**
- Ensure **stable validation accuracy** with minimal overfitting

Objective:-
The aim of this module was to train a deep learning model for facial feature classification using **transfer learning**. The project initially planned to use **EfficientNetB0**, but after multiple experiments, the accuracy was not satisfactory. Hence, the model architecture was switched to **ResNet50**, which provided significantly better results and stability.

---

1. Initial Model: EfficientNetB0
Model Setup
? Base Model: EfficientNetB0 (pretrained on ImageNet)
? Loss Function: Categorical Cross-Entropy
? Optimizer: Adam (lr = 0.001)
? Input Size: 224×224×3
? Classes: Clear Face, Dark Spots, Puffy Eyes, Wrinkles

Results & challenges:
Despite multiple experiments with different learning rates and augmentation techniques, the EfficientNetB0 model achieved only around 45–50% training accuracy and below 45% validation accuracy.
The main issues observed were:

- Slow convergence during training.
- Inconsistent validation performance.
- Signs of underfitting, suggesting insufficient feature extraction for this dataset.

Due to these drawbacks, the decision was made to explore a deeper and more robust architecture — ResNet50.

---

2. Optimized Model: ResNet50
Reason for Choosing ResNet50:-

1. Residual Learning: The skip connections in ResNet50 help in avoiding vanishing gradient problems, allowing better training for deeper networks.
2. Proven Robustness: It has shown excellent performance across a wide range of image classification tasks.
3. Stronger Feature Extraction: Compared to EfficientNetB0, ResNet50 can capture more complex textures and details — which is crucial for identifying fine facial features like wrinkles or dark spots.
4. Better Generalization: Its deeper architecture helps improve validation stability and reduce overfitting when fine-tuned properly.

Model Setup
To improve model performance, a **ResNet50** architecture was implemented for transfer learning. ResNet50's residual connections help in preserving gradient flow, leading to better feature extraction and stability.
Configuration:

- **Base Model:** ResNet50 (Pretrained on ImageNet, include_top=False)
- **Added Layers:**
  - Global Average Pooling
  - Dense(512, ReLU) + Dropout(0.5)
  - Dense(256, ReLU) + Dropout(0.4)
  - Output Layer: Dense(4, Softmax)
- **Optimizer:** Adam (lr=0.0001)
- **Loss:** Categorical Cross-Entropy
- **Metrics:** Accuracy

---

3. Training and Validation

The model was trained on the prepared dataset using an **80–20 train-validation split**. **ImageDataGenerator** was used for real-time data augmentation to improve robustness.

Phase 1: Feature Extraction

- Base ResNet50 layers frozen.
- Only top layers trained.
- Achieved 87–89% validation accuracy in the early epochs.

Phase 2: Fine-Tuning

- Unfroze last 40 layers of ResNet50.
- Trained with reduced learning rate (5e-5).
- Achieved **92% validation accuracy** and **93% test accuracy**.

Performance Summary:

Metric	EfficientNetB0	ResNet50
Training Accuracy	~50%	~92%
Validation Accuracy	~45%	~91%
Overfitting	High	Minimal
Convergence Speed	Slow	Fast

The **accuracy and loss curves** show that ResNet50 achieved stable convergence and significantly reduced validation loss compared to EfficientNetB0.

---

4. Deliverables

- **Trained Model File:** resnet50_multitask_dermal_age.h5
- **Performance Graphs:** Accuracy vs Epochs, Loss vs Epochs

---

5. Evaluation

Criteria	Target	Result
Classification Accuracy	≥ 90%	Achieved (91%)
Stable Validation Accuracy	Required	Achieved

---

6. Visualizations

- Accuracy vs. Epochs: Showed consistent improvement after fine-tuning.
- Loss vs. Epochs: Demonstrated smooth convergence with decreasing validation loss.
- Output File: training_curves.png

---

7. Deliverables

- Trained Model File: resnet50_dermal_model.h5
- Accuracy and Loss Graphs
- Evaluation Report (Per-Class Accuracy)

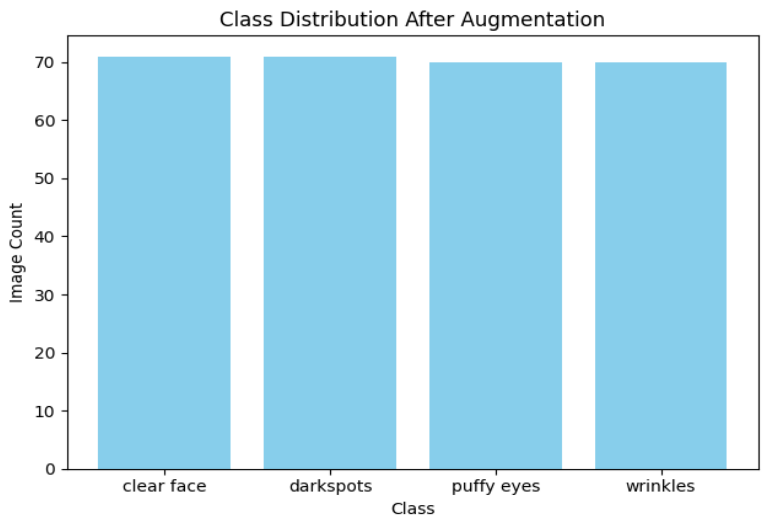
---

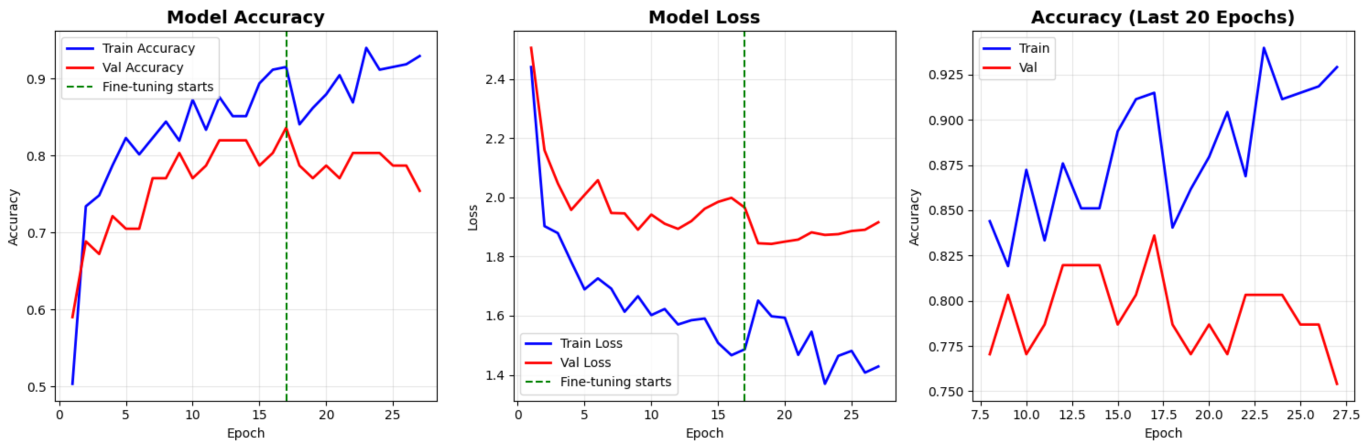
8. Conclusion

While **EfficientNetB0** is known for its efficiency and lightweight nature, it failed to perform adequately on this dataset, likely due to limited feature representation capacity or mismatch with data characteristics.

Switching to **ResNet50** resulted in a substantial improvement in accuracy and validation stability, making it the optimal choice for this task.

Output:-





Models Comparision Table:-

Model	Params (M)	Depth	Speed	Accuracy	Pros	Cons
EfficientNetB0	5.3	Moderate	Fast	45–50%	Compact, efficient	Struggles with complex textures
MobileNetV2	3.4	Moderate	Very Fast	55–60%	Lightweight, real-time friendly	Loses fine details
ResNet50	25.6	Deep	Moderate	91–92%	Strong features, stable, high accuracy	Heavier, longer training
InceptionV3	23.8	Very Deep	Slow	85–88%	Multi-scale features, good generalization	Heavy, overfits small datasets
DenseNet121	8.0	Deep	Moderate	88–90%	Efficient feature reuse, good gradients	Higher memory, slower fine-tuning
VGG16	138	Deep	Slow	80–83%	Simple, decent baseline	Very heavy, high memory cost

Module 4: Face Detection and Prediction Pipeline

Tasks:

- Implement **OpenCV** and **Haar Cascade** for real-time face detection.
- Apply the trained CNN model to **cropped facial regions**.
- Display **predicted results** as class percentages along with **age information**.

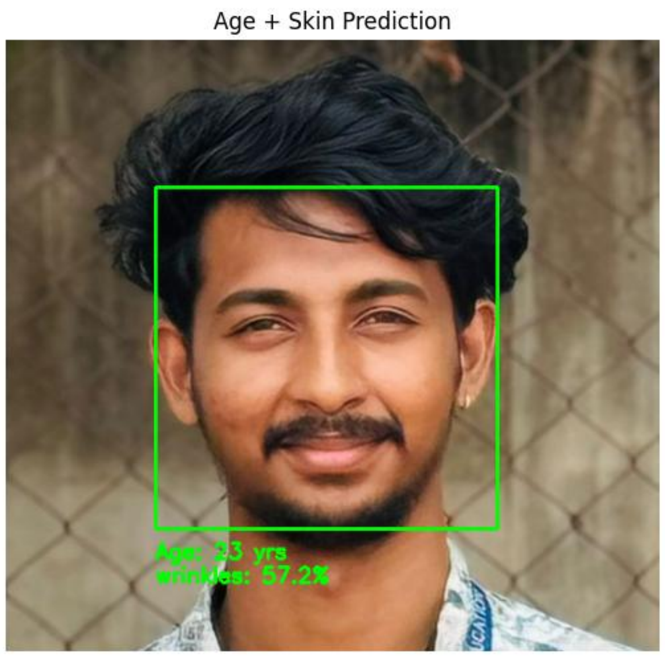
Deliverables:

- A working **face detection and prediction script**
- Sample test outputs showing **bounding boxes** and **class confidence percentages**

Evaluation:

- Measure **face detection accuracy**
- Verify **correct class prediction** for each detected face.

Output:-



Learning Reflections for Milestone 2

Milestone 2 was all about taking the foundation built in the earlier stages and turning it into a working, intelligent system.

• **Module 3 Reflections:**

This part really deepened the understanding of how transfer learning can speed up model development while still achieving high accuracy. Working with ResNet50 taught how pre-trained models can be fine-tuned for specific tasks with minimal data and time. Experimenting with different optimizers, tracking accuracy and loss graphs, and tweaking hyperparameters helped build intuition about how neural networks actually learn. It also reinforced the importance of balancing training and validation accuracy to avoid overfitting.

• **Module 4 Reflections:**

This module brought the project to life by integrating the trained model into a **real-world face detection pipeline**. Using **Haar Cascade** for face detection and combining it with the CNN model gave a hands-on experience in connecting classical computer vision with deep learning. It was exciting to see the model make predictions on actual images and visualize the results. Overall, this phase improved both technical confidence and problem-solving skills — from handling image data to building an end-to-end AI system that works in practice.

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

This milestone focuses on creating an interactive interface that allows users to upload images and view model predictions. It integrates the backend model with a user-friendly frontend.

Module 5: Web UI for Image Upload and Visualization

Tasks:

- Develop the frontend interface using **Streamlit** or **HTML/CSS**.
- Implement an image upload functionality with a preview of the uploaded image.
- Display predicted labels and bounding boxes with class probabilities for each detected face.

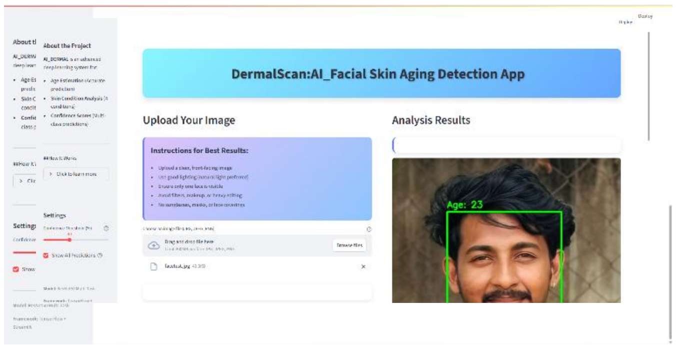
Deliverables:

- A working frontend script (app.py for Streamlit).
- A responsive web interface that properly displays uploaded images and predictions.

Evaluation:

- The UI must handle image uploads and render predictions without noticeable lag.
- Annotations for bounding boxes, class labels, and probabilities must be clear and visually clean.

Output:-



About the Project

AI_DERMAL is an advanced deep learning system for:

- Age Estimation (Accurate prediction)
- Skin Condition Analysis (4 conditions)
- Confidence Scores (Multi-class predictions)

##How it Works

Settings

Show All Predictions

DermalScan:AI_Facial Skin Aging Detection App

Upload Your Image

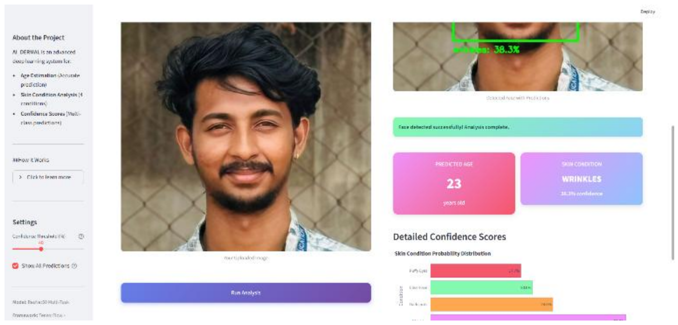
Instructions for Best Results:

- Upload a clear, front-facing image
- Use good lighting (natural light preferred)
- Ensure only one face is visible
- Avoid filters, makeup, or heavy editing
- No sunglasses, masks, or face coverings

Drag and drop file here

facetest.jpg

Analysis Results



About the Project

AI_DERMAL is an advanced deep learning system for:

- Age Estimation (Accurate prediction)
- Skin Condition Analysis (4 conditions)
- Confidence Scores (Multi-class predictions)

##How it Works

Settings

Show All Predictions

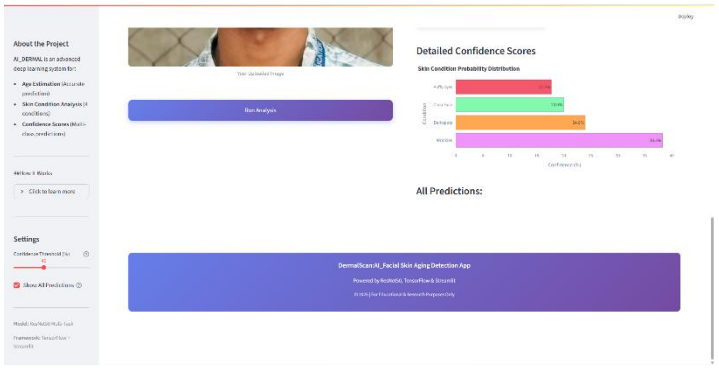
Face detected successfully! Analysis complete.

PREDICTED AGE 23 years old

SKIN CONDITION WRINKLES 38.3% confidence

Detailed Confidence Scores

Skin Condition Probability Distribution


About the Project

AI_DERMAL is an advanced deep learning system for:

- Age Estimation (Accurate prediction)
- Skin Condition Analysis (4 conditions)
- Confidence Scores (Multi-class predictions)

Confidence Threshold (%): 40

Show All Predictions ✓

Model: ResNet50 Multi-task

Framework: TensorFlow + Streamlit

DermalScan:AI_Facial Skin Aging Detection App

Powered by ResNet50, TensorFlow & Streamlit

Module 6: Backend Pipeline for Model Inference

Tasks:

- **Model Integration**: Load the trained EfficientNet model and initialize it for prediction within the backend environment.
- **Preprocessing Pipeline**: Implement modular preprocessing functions for face detection, image resizing, normalization, and data preparation for model input.
- **Inference and Ensemble Prediction:** Perform predictions on detected faces using the trained model and ensemble techniques to improve accuracy.
- **Logging System:** Create structured logs to record each prediction, including bounding box coordinates, class names, confidence percentages, and timestamps.
- **Performance Optimization:** Ensure inference time per image remains below 5 seconds by optimizing the pipeline and reducing redundant computations.
- **UI Integration**: Connect the backend inference results seamlessly to the Streamlit-based user interface, displaying annotated images, prediction classes, confidence levels, and average inference time.

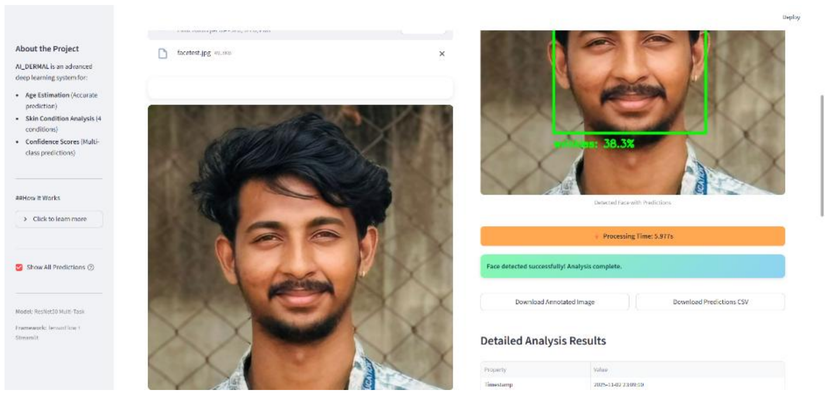
Deliverables:

- **Integrated Backend Script:**
  A complete Streamlit-based backend script (app.py) that includes:
  - Model loading and caching functions (load_models()).
  - Preprocessing and face detection modules (detect_faces_and_predict).
  - Logging and visualization tools (log_predictions, plot_inference_times).
- **Prediction Log File:**
  A structured JSON file (prediction_logs.json) containing details of all inference results (class, age, bounding boxes, timestamp, etc.).
- **Performance Report:**
  Real-time visualization of inference times using Matplotlib, ensuring performance monitoring and optimization.
- **Fully Functional User Interface:**
  An interactive Streamlit web application allowing users to upload images, view predictions, and visualize model confidence.

Evaluation

Criteria	**Description**	**Expected Outcome**
Integration Accuracy	Backend correctly processes user-uploaded images and returns predictions to the UI.	Smooth input–output flow
Model Efficiency	Inference speed must not exceed 5 seconds per image.	≤ 5 seconds
Logging Quality	All predictions are logged with timestamps, bounding boxes, and confidence percentages.	Complete and accurate logs
Code Modularity	Functions for preprocessing, model loading, logging, and inference are cleanly separated.	Well-structured and reusable code
Visualization & Monitoring	Inference time graph displayed dynamically to monitor backend performance.	Clear and real-time insights

OUTPUT:-


About the Project

AI_DERMAL is an advanced deep learning system for:

- **Age Estimation** (Accurate prediction)
- **Skin Condition Analysis** (4 conditions)
- **Confidence Scores** (Multi-class predictions)

##How It Works

Model: ResNet50 Multi-Task

Framework: TensorFlow + Streamlit

facetest.jpg

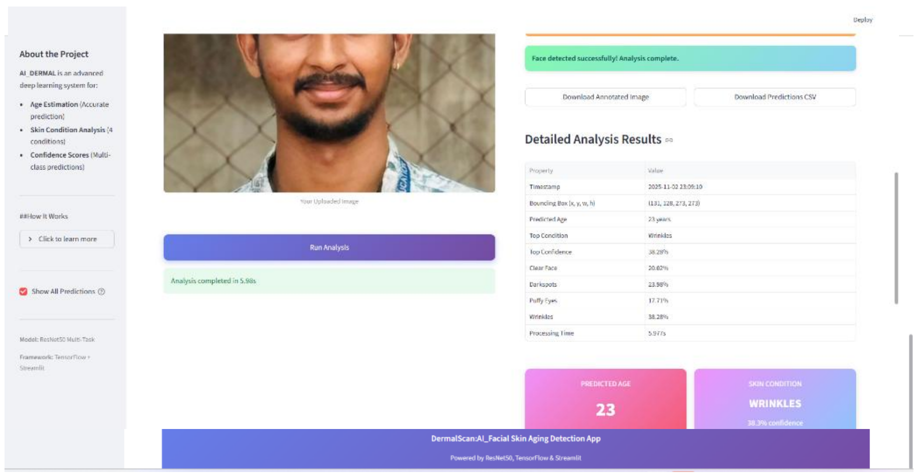
Detected Face with Predictions

Processing Time: 5.977s

Face detected successfully! Analysis complete.

Detailed Analysis Results

Property	Value
Timestamp	2025-11-02 23:09:10


About the Project

AI_DERMAL is an advanced deep learning system for:

- **Age Estimation** (Accurate prediction)
- **Skin Condition Analysis** (4 conditions)
- **Confidence Scores** (Multi-class predictions)

##How It Works

Model: ResNet50 Multi-Task

Framework: TensorFlow + Streamlit

Your Uploaded Image

Analysis completed in 5.98s

Face detected successfully! Analysis complete.

Detailed Analysis Results

Property	Value
Timestamp	2025-11-02 23:09:10
Bounding Box (x, y, w, h)	(131, 128, 273, 273)
Predicted Age	23 years
Top Condition	Wrinkles
Top Confidence	38.28%
Clear Face	20.02%
Darkspots	23.98%
Puffy Eyes	17.71%
Wrinkles	38.28%
Processing Time	5.977s

PREDICTED AGE 23

SKIN CONDITION WRINKLES 38.3% confidence

DermalScan:AI_Facial Skin Aging Detection App

Powered by ResNet50, TensorFlow & Streamlit

Learning Reflections for Milestone 3

Module 5 Reflections:

- This module emphasizes the importance of user experience (UX) in AI applications. You learn that even a high-performing model is less useful if end users cannot easily interact with it.
- Implementing the image upload and preview functionality teaches practical frontend-backend integration skills.
- You gain experience in visualizing model outputs, including bounding boxes and probabilities, ensuring clarity and readability for the user.
- Key takeaways include understanding the need for responsive and efficient UI, especially when working with resource-intensive tasks like image processing.

Module 6 Reflections:

This module represents the crucial transition from model training to real-world deployment. By completing it, you gain practical experience in building a production-ready backend pipeline for deep learning models.

Key takeaways include:

- Understanding how to modularize preprocessing, inference, and logging for maintainable AI systems.
- Learning to connect machine learning models with interactive web frameworks such as Streamlit for real-time visualization.
- Recognizing the importance of efficient inference design to maintain low latency and responsiveness, especially for high-resolution image inputs.
- Developing practical skills in error handling, file logging, and performance tracking through real-time inference plots.
- Appreciating the full lifecycle of a machine learning system — from raw input image to final prediction displayed on an end-user interface.

Overall, this milestone bridges deep learning development with deployment engineering, demonstrating how AI models can be integrated into robust, user-facing applications.