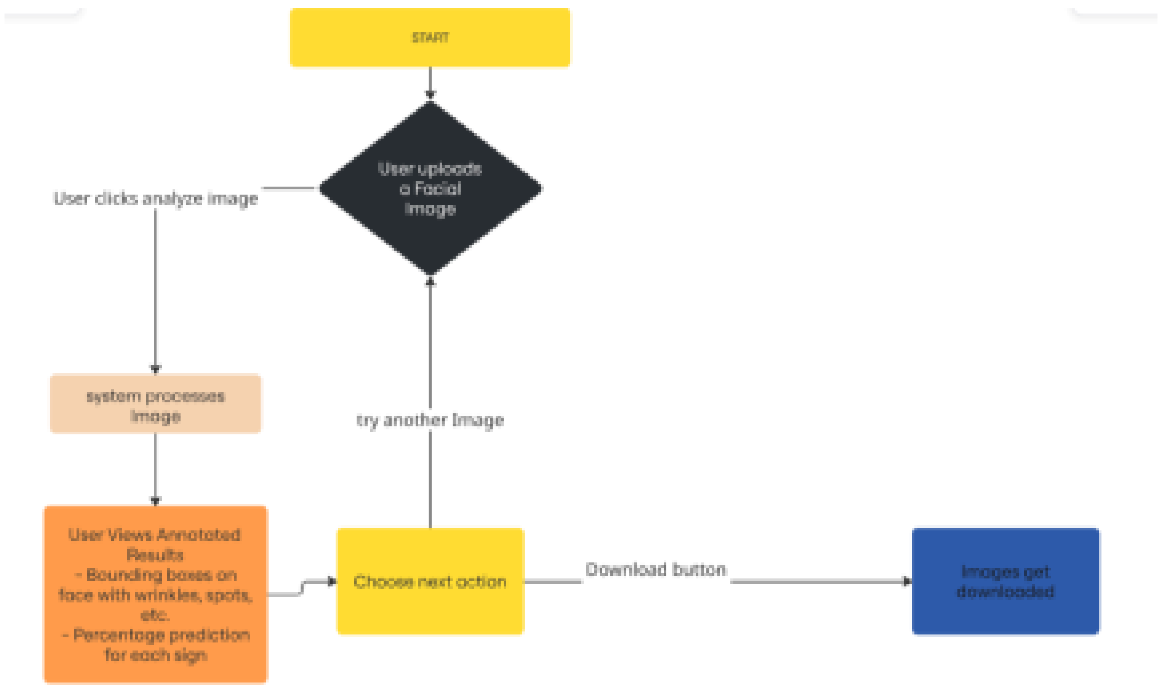
DermalScan:AI_Facial Skin Aging Detection App

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.


START
User uploads a Facial Image
User clicks analyze image
system processes Image
try another Image
User Views Annotated Results
- Bounding boxes on face with wrinkles, spots, etc.
- Percentage prediction for each sign
Choose next action
Download button
Images get downloaded

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 Preparation and Preprocessing (Weeks 1–2)

Module 1: Dataset Setup and Image Labeling

Tasks:
- Set up and inspect dataset of facial images or create own dataset which you can categorize or use the dataset:
https://drive.google.com/drive/folders/1HtKbXujonS0jVNOKg0qzzAdDvCaGDd1A?usp=sharing
- Label classes: wrinkles, dark spots, puffy eyes, clear skin.
- Ensure balanced distribution and clean samples.

Deliverables:
- Cleaned and labeled dataset
- Class distribution plot

Evaluation:
- Proper class balance
- Accurate labeling and inspection

Module 2: Image Preprocessing and Augmentation

Tasks:
- Resize and normalize images (224x224).
- Apply image augmentation (flip, rotation, zoom).
- Encode class labels using one-hot encoding.

Deliverables:
- Preprocessed and augmented dataset
- Augmentation script with visualization

Evaluation:
- Augmentation quality and dataset readiness
- Class diversity retained post-augmentation

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

Module 3: Model Training with EfficientNetB0

Tasks:
- Use pretrained EfficientNetB0 for transfer learning.
- Train with categorical cross-entropy loss and Adam optimizer.
- Validate model and plot training metrics.

Deliverables:
- Trained CNN model (.h5 file)
- Accuracy and loss curves

Evaluation:
- ≥ 90% classification accuracy
- Stable validation accuracy

Module 4: Face Detection and Prediction Pipeline
Tasks:
- Use OpenCV and Haar Cascade for face detection.
- Apply model to cropped face regions.
- Display predictions as percentages along with age.
Deliverables:
- Face detection and prediction script
- Test output with bounding boxes and percentages
Evaluation:
- Face detection accuracy
- Correct class prediction

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

Module 5: Web UI for Image Upload and Visualization
Tasks:
- Develop frontend using Streamlit or HTML/CSS.
- Implement image upload field and output preview.
- Display labels and bounding boxes with class probability.
Deliverables:
- Frontend app.py or HTML/CSS script
- Responsive web interface
Evaluation:
- No UI lag on upload or render
- Clean annotation visualization

Module 6: Backend Pipeline for Model Inference
Tasks:
- Modularize inference and preprocessing code.
- Load EfficientNet model and return results to UI.
- Log predictions and bounding box data.
Deliverables:
- Integrated backend script
- End-to-end testing with UI
Evaluation:
- Seamless input-to-output flow
- ≤ 5 seconds per image

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

Module 7: Export and Logging
Tasks:
- Allow download of annotated image and CSV predictions.
- Run testing on diverse images and finalize results.
Deliverables:
- Export option added to frontend
- Final result logs and annotated outputs

Evaluation:
- Accurate export and log consistency
- Proper CSV formatting

Module 8: Documentation and Final Presentation

Tasks:
- Create user and developer guides (README).
- Prepare GitHub repo, slides, and walkthrough video.

Deliverables:
- Final documentation and GitHub project
- Presentation slides and video (optional)

Evaluation:
- Clear documentation structure
- Demo-ready output

Evaluation Criteria:

Milestone	Focus Area	Metric / Evaluation Method	Target / Goal
Milestone 1	Data Preparation & Preprocessing	Dataset quality, augmentation effectiveness	Balanced & clean dataset
Milestone 2	Model Performance	Accuracy & loss metrics	good test accuracy
Milestone 3	UI & Backend	Upload-to-output time & usability	≤ 5s per image
Milestone 4	Final Delivery	Export functionality & documentation	Complete & professional

Tech Stack:

Area	Tools / Libraries
Image Ops	OpenCV, NumPy, Haarcascade
Model	TensorFlow/Keras, EfficientNetB0
Dataset	Labeled facial images dataset
Frontend	Streamlit or HTML, CSS

Backend	Python, Modularized Inference
Evaluation	Accuracy, Loss, Confusion Matrix
Exporting	CSV, Annotated Image, PDF (opt.)