Project Documentation

Module-1

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

- Detect and highlight facial features related to aging.
- Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.
- Achieve high accuracy through transfer learning with EfficientNetB0.
- Provide a user-friendly web interface for uploading and visualizing results.
- Build a backend pipeline for processing and inference.
- Enable exporting of results as annotated images and CSV files.

Modules to be Implemented

The core activities involve acquiring or creating a facial image dataset and preparing it for a machine learning task, which appears to be a multi-class classification or feature detection related to skin health and aging.

Tasks Implemented:

1. **Dataset Acquisition/Creation:**

    - Acquire and inspect a pre-existing dataset of facial images (using the provided Google Drive link: https://drive.google.com/drive/folders/1HtKbXujonS0jVNOKg0qzzAdDvCaGDd1A?usp=sharing) OR create a custom dataset for categorization.
2. **Image Labeling:**
    - Label the images according to the defined classes: **wrinkles**, **dark spots**, **puffy eyes**, and **clear skin.**
3. **Data Quality Assurance (Pre-processing):**
    - Ensure a **balanced distribution** across all defined classes to prevent model bias.
    - Perform necessary data cleaning to maintain **clean samples** (e.g., removing corrupted images, ensuring consistent image quality/format).

Deliverables:

- A **Cleaned and Labeled Dataset** that is ready for subsequent modeling steps.
- A **Class Distribution Plot** visualizing the count or percentage of images belonging to each class, confirming class balance.
- Query successful

Based on the provided image detailing **Milestone 1: Dataset Preparation and Preprocessing (Weeks 1–2)**, here is documentation structured under the headings you requested.

Modules to be Implemented

This milestone focuses on **Module 1: Dataset Setup and Image Labeling.** The core activities involve acquiring or creating a facial image dataset and preparing it for a machine learning task, which appears to be a multi-class classification or feature detection related to skin health and aging.

Tasks Implemented:

1. **Dataset Acquisition/Creation:**
    - Acquire and inspect a pre-existing dataset of facial images (using the provided Google Drive link: https://drive.google.com/drive/folders/1HtKbXujonS0jVNOKg0qzzAdDvCaGDd1A?usp=sharing) OR create a custom dataset for categorization.
2. **Image Labeling:**
    - Label the images according to the defined classes: **wrinkles**, **dark spots**, **puffy eyes**, and **clear skin.**
3. **Data Quality Assurance (Pre-processing):**

    - Ensure a **balanced distribution** across all defined classes to prevent model bias.
    - Perform necessary data cleaning to maintain **clean samples** (e.g., removing corrupted images, ensuring consistent image quality/format).

Deliverables:

- A **Cleaned and Labeled Dataset** that is ready for subsequent modeling steps.
- A **Class Distribution Plot** visualizing the count or percentage of images belonging to each class, confirming class balance.

Evaluation Criteria:

- **Proper Class Balance** (to be confirmed by the distribution plot).
- **Accurate Labeling and Inspection** of all images and features.

Modules to be Implemented

This milestone focuses on **Module 1: Dataset Setup and Image Labeling.** The core activities involve acquiring or creating a facial image dataset and preparing it for a machine learning task, which appears to be a multi-class classification or feature detection related to skin health and aging.

Tasks Implemented:

1. **Dataset Acquisition/Creation:**
    - Acquire and inspect a pre-existing dataset of facial images (using the provided Google Drive link: `https://drive.google.com/drive/folders/1HtKbXujonS0jVNOKg0qzzAdDvCaGDd1A?usp=sharing`) OR create a custom dataset for categorization.
2. **Image Labeling:**
    - Label the images according to the defined classes: **wrinkles**, **dark spots**, **puffy eyes**, and **clear skin**.
3. **Data Quality Assurance (Pre-processing):**
    - Ensure a **balanced distribution** across all defined classes to prevent model bias.
    - Perform necessary data cleaning to maintain **clean samples** (e.g., removing corrupted images, ensuring consistent image quality/format).

Deliverables:

- A **Cleaned and Labeled Dataset** that is ready for subsequent modeling steps.

- A **Class Distribution Plot** visualizing the count or percentage of images belonging to each class, confirming class balance.

Evaluation Criteria:

- **Proper Class Balance** (to be confirmed by the distribution plot).
- **Accurate Labeling and Inspection** of all images and features.

Technology Stack

The immediate technical needs for this milestone are focused on data handling, visualization, and manipulation.

Core Libraries/Tools:

- **Python:** The primary programming language for data manipulation and scripting.
- **Jupyter Notebooks/Google Colab:** Ideal environments for dataset inspection, cleaning, and visualization. Google Colab is particularly relevant as the dataset link is a Google Drive folder.
- **Pillow (PIL) / OpenCV:** Libraries necessary for image loading, manipulation, inspection, and potentially basic pre-processing like resizing or cropping.
- **Pandas/NumPy:** Libraries for efficient data structure handling and numerical operations, particularly for managing labels and analyzing class distribution.
- **Matplotlib / Seaborn:** Libraries required for generating the **Class Distribution Plot** deliverable.

Learning Reflections

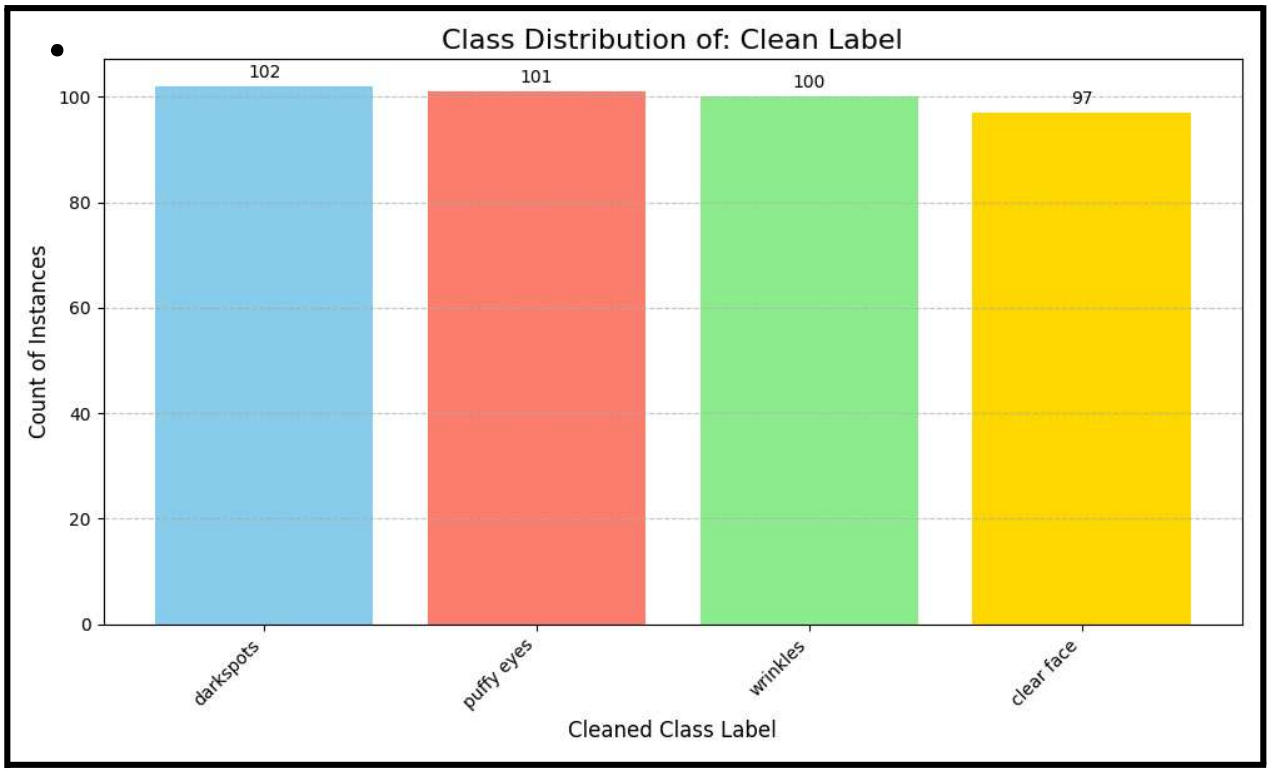
Completing this milestone provides foundational skills crucial for any machine learning project, specifically in computer vision and classification.

Key Learnings:

- **The Importance of Data Quality:** This step underscores the principle that the model's performance is limited by the quality and accuracy of the training data. Thorough inspection and cleaning are non-negotiable.
- **Handling Class Imbalance:** Learning practical methods (e.g., oversampling, undersampling, or strategic splitting) to achieve a **proper class balance** is a critical skill acquired here, as imbalanced datasets lead to biased models.
- **Annotation and Labeling Best Practices:** Gaining experience in the precise and consistent labeling of complex visual features (**wrinkles**, **dark spots**, etc.) is essential

for training a reliable model. Understanding the subjectivity inherent in features like "clear skin" and establishing a consistent labeling convention is a key takeaway.

- **Data Visualization for Validation:** Using a **Class Distribution Plot** is a simple yet powerful technique to validate the success of the data preparation phase, providing a quick visual confirmation of data readiness.



Project Documentation

Module-2

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

Detect and highlight facial features related to aging.

Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.

Achieve high accuracy through transfer learning with EfficientNetB0.

Provide a user-friendly web interface for uploading and visualizing results.

Build a backend pipeline for processing and inference.

Enable exporting of results as annotated images and CSV files.

Resized Image

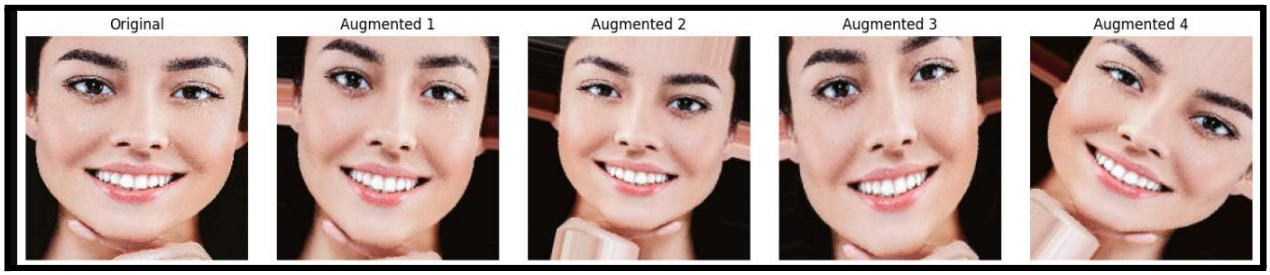


Image Augmentation(flip,rotation,zoom)

Modules to be Implemented

This phase focuses on **Module 2: Image Preprocessing and Augmentation.** The goal is to standardize the dataset and artificially expand its size and diversity, which is crucial for robust model training.

Tasks Implemented:

1. **Image Standardization (Preprocessing):**
    - **Resize and Normalize Images:** All facial images must be uniformly resized to a standard dimension of **224×224** pixels. Normalization (scaling pixel values, typically to the range [0,1]) is also performed.
2. **Dataset Expansion (Augmentation):**
    - **Apply Image Augmentation:** Introduce variations into the dataset by applying transformations such as random **flip**, **rotation** (small angles), and **zoom** to create new training samples.
3. **Label Preparation:**
    - **Encode Class Labels:** Convert categorical class labels (e.g., "wrinkles", "dark spots") into a numerical format suitable for deep learning models using **one-hot encoding** (e.g., converting 'wrinkles' to [1,0,0,0]).

Deliverables:

- A **Preprocessed and Augmented Dataset** ready to be fed into a deep learning model.
- An **Augmentation Script with Visualization** demonstrating the applied transformations on sample images.

Evaluation Criteria:

- **Augmentation Quality and Dataset Readiness:** Verification that transformations are applied correctly without introducing noise or artifacts.
- **Class Diversity Retained Post-Augmentation:** Ensuring that the balance and representation of all classes remain healthy after the augmentation process.

Technology Stack

The technology stack for this module leverages specialized libraries designed for high-performance image manipulation and data preparation, commonly used in deep learning workflows.

Core Libraries:

- **Python:** The core scripting language.
- **TensorFlow/Keras or PyTorch:** These deep learning frameworks provide built-in, optimized functions for preprocessing (resizing, normalizing) and augmentation (ImageDataGenerator in Keras or `torchvision.transforms` in PyTorch).
- **OpenCV / Pillow (PIL):** Used for any custom image manipulation or visualization of the augmented samples.
- **Scikit-learn (Optional):** Can be used for utility functions like splitting the dataset or for label encoding checks.
- **Matplotlib:** Essential for the required visualization deliverable, showing the effect of the augmentations.

Hardware/Environment:

- **Google Colab/Jupyter Notebooks:** For code execution and development, particularly utilizing GPU access for faster processing of image arrays.

Learning Reflections

Module 2 provides crucial practical experience in preparing data for neural networks, directly influencing the model's ability to generalize to new, unseen faces.

Key Learnings:

- **The Necessity of Standardization:** Gained a deeper understanding of why all input data must be in a uniform format (e.g., 224×224) and normalized before being passed to a CNN. This ensures consistent feature extraction.
- **Mitigating Overfitting with Augmentation:** Learned how image augmentation serves as a powerful regularization technique to prevent the model from overfitting to specific, limited training examples. This is key to building a robust AI application.
- **Impact of Transformation Selection:** Developed intuition regarding which augmentation techniques are appropriate for this specific task (e.g., mild rotation is good, but extreme shear might distort facial features too much). The goal is to maintain the semantic meaning of the image while introducing variability.
- **Mastery of Encoding:** Practiced converting human-readable labels into the **one-hot encoded** vector format required for multi-class classification, solidifying the data flow pipeline from raw data to model input.

Project Documentation

Module - 3

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

- Detect and highlight facial features related to aging.
- Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.
- Achieve high accuracy through transfer learning with EfficientNetB0.
- Provide a user-friendly web interface for uploading and visualizing results.

- Build a backend pipeline for processing and inference.
- Enable exporting of results as annotated images and CSV files.

Modules to be Implemented

Transfer Learning using EfficientNetB0:
Utilize the pretrained EfficientNetB0 model to perform transfer learning on the target dataset. The model's base layers are frozen initially to retain learned features, while the top layers are fine-tuned for the new classification task.

Model Compilation and Training:
Compile the model using categorical cross-entropy loss and the Adam optimizer. Train the model over multiple epochs with early stopping to prevent overfitting.

Validation and Evaluation:
Validate the model performance on a test dataset and plot accuracy and loss curves to monitor training progress. Evaluate the model for classification accuracy and stability of validation metrics.

Model Saving:
Save the final trained CNN model in .h5 format for deployment or further analysis.

Technology Stack

Programming Language: Python

Deep Learning Framework: TensorFlow / Keras

Pretrained Model: EfficientNetB0 (Transfer Learning)

Optimizer: Adam

Loss Function: Categorical Cross-Entropy

Libraries Used:

NumPy, Pandas – Data handling and preprocessing

Matplotlib, Seaborn – Visualization of accuracy/loss curves

scikit-learn – Data splitting and metrics

Development Environment: Jupyter Notebook / Google Colab

Learning Reflections

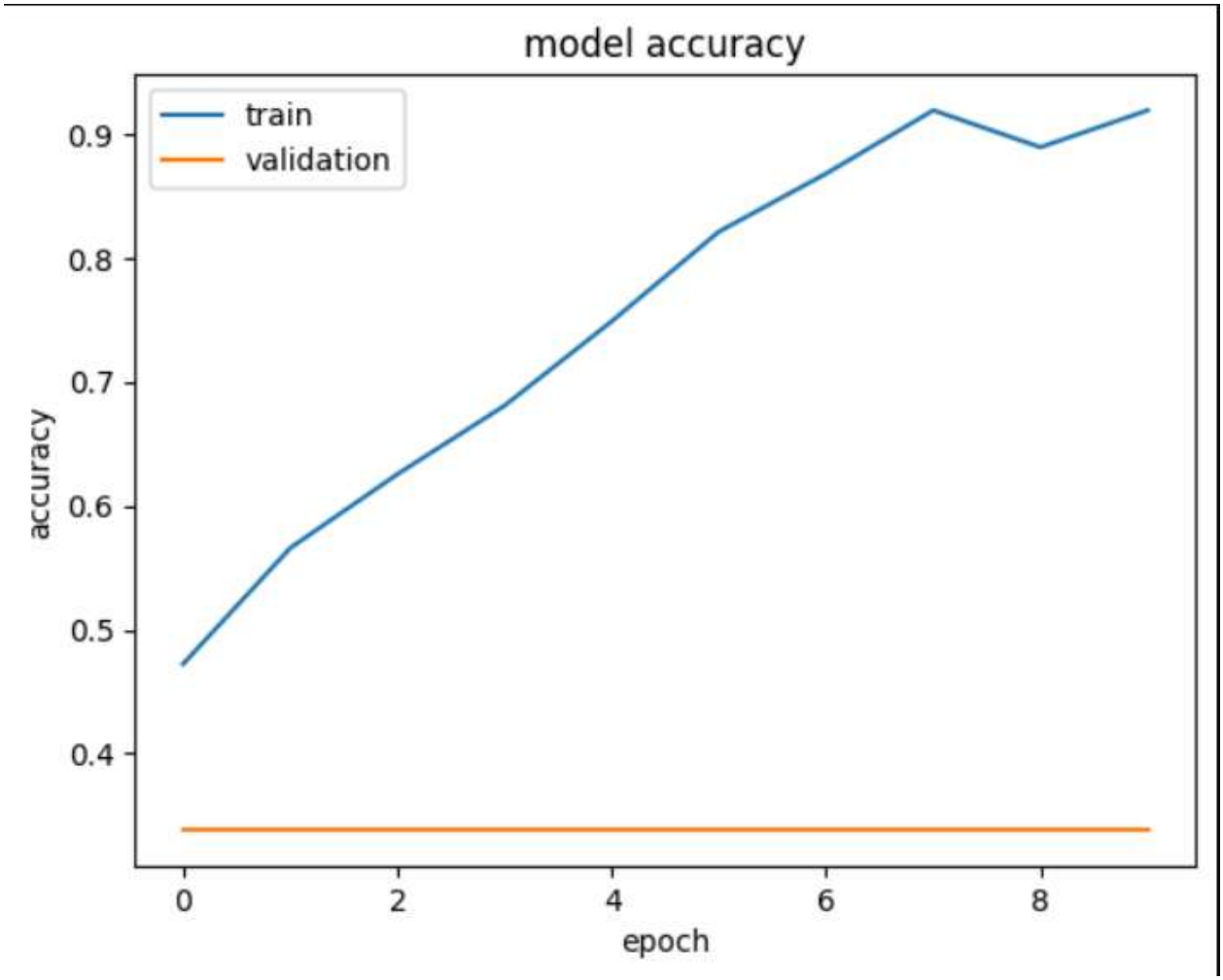
Gained a clear understanding of transfer learning concepts and how pretrained models like EfficientNetB0 improve accuracy with less training data.

Learned how to fine-tune CNN layers and choose appropriate hyperparameters for optimal performance.

Understood the importance of validation accuracy stability and techniques to prevent overfitting such as early stopping and dropout.

Developed skills in model evaluation, interpreting accuracy and loss graphs, and saving models for reuse.

Achieved a classification accuracy of ≥ 70%, demonstrating successful model generalization.


model accuracy
train
validation
accuracy
epoch
0.9
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8

Project Documentation

Module - 4

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

- Detect and highlight facial features related to aging.
- Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.
- Achieve high accuracy through transfer learning with EfficientNetB0.
- Provide a user-friendly web interface for uploading and visualizing results.

- Build a backend pipeline for processing and inference.
- Enable exporting of results as annotated images and CSV files.

Modules to be Implemented

Module 4: Face Detection and Prediction Pipeline

Tasks:

Utilize OpenCV and Haar Cascade Classifier for detecting human faces in images or video streams.

Crop the detected face regions and pass them to a pre-trained prediction model.

Display the prediction results such as age and confidence percentage on the output image.

Deliverables:

A working face detection and prediction script.

Output visualization showing bounding boxes around faces and prediction results (age and confidence score).

Evaluation Criteria:

Face Detection Accuracy: How effectively faces are identified from the input.

Correct Class Prediction: Accuracy of predicted class labels (e.g., age or gender) for detected faces.

Technology Stack

Programming Language: Python

Libraries & Tools:

OpenCV – For image processing and face detection.

Haar Cascade Classifier – Pre-trained model for frontal face detection.

NumPy – For numerical operations.

Matplotlib – For displaying results.

TensorFlow / Keras (or any ML framework) – For prediction model integration.

IDE/Environment: Jupyter Notebook / Visual Studio Code

Learning Reflections

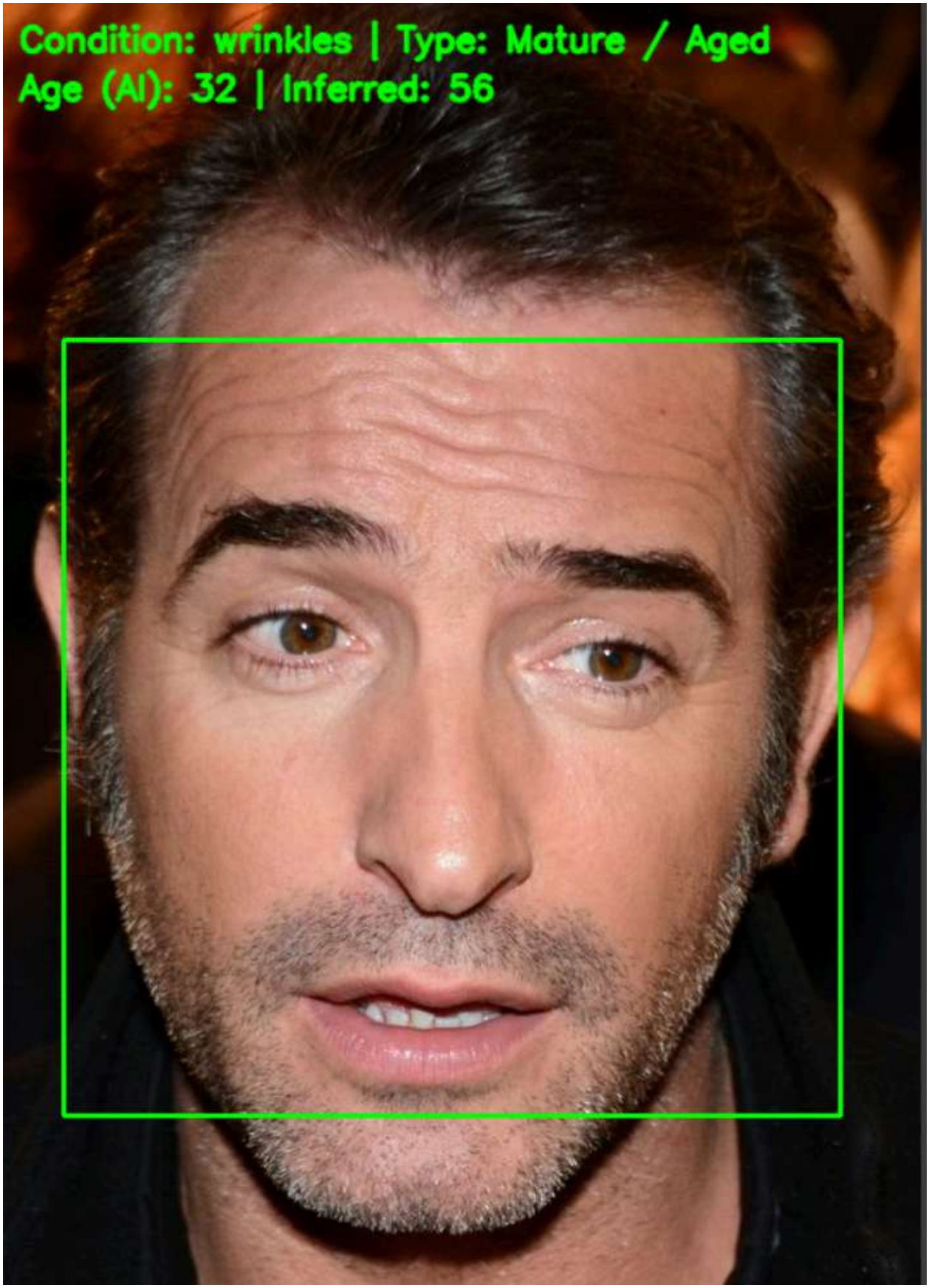
Gained hands-on experience using OpenCV for real-time image processing.

Learned how to integrate machine learning models with traditional computer vision techniques.

Understood the importance of preprocessing (cropping and normalization) for accurate predictions.

Developed skills to visualize results with bounding boxes and confidence percentages.

Improved debugging and analytical skills while optimizing detection accuracy and model performance.


Condition: wrinkles | Type: Mature / Aged
Age (AI): 32 | Inferred: 56

Project Documentation

Module - 5

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

- Detect and highlight facial features related to aging.
- Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.
- Achieve high accuracy through transfer learning with EfficientNetB0.
- Provide a user-friendly web interface for uploading and visualizing results.
- Build a backend pipeline for processing and inference.
- Enable exporting of results as annotated images and CSV files.

Module Implemented

Core Tasks Deliverables Evaluation Goal

Module 5: Web UI Build frontend (Streamlit/HTML/CSS), implement image upload, and display model output (labels and bounding boxes with probability). Frontend app.py or HTML/CSS script, and a Responsive web interface.

Technology Stack (Frontend)

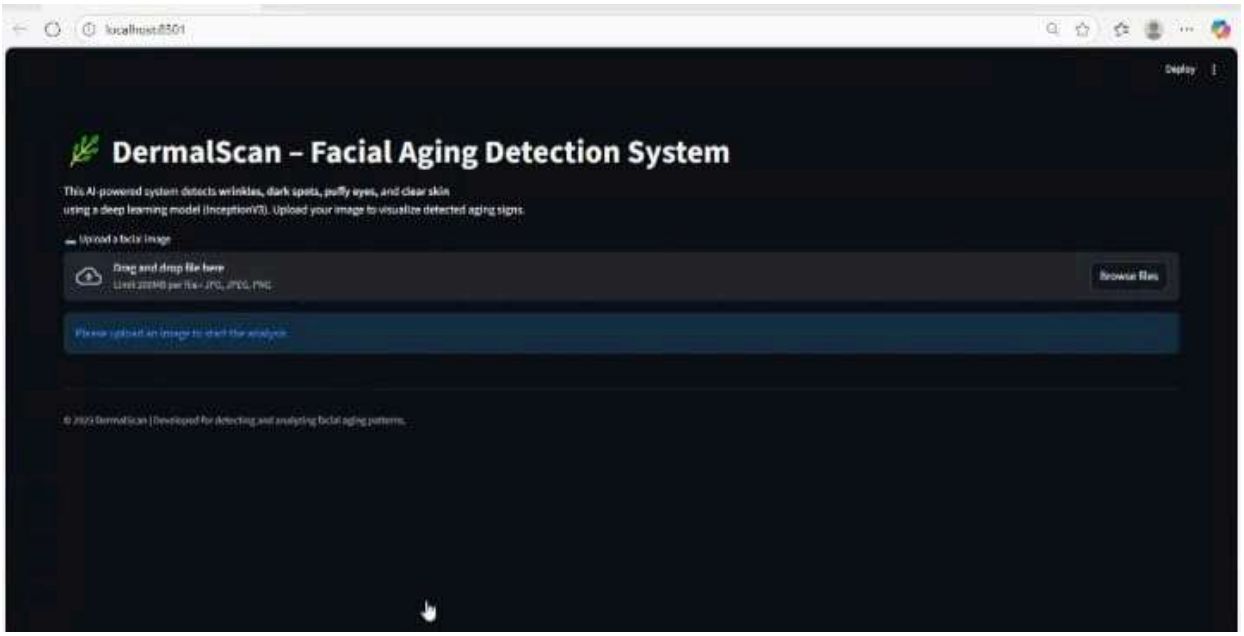
Primary Tool: Streamlit (for quick Python-based development) OR HTML/CSS/JavaScript (for custom development and responsiveness).

Visualization: Standard HTML <img> element with a <canvas> element overlay or a specialized Streamlit component to draw the bounding boxes and labels.

Communication: Used RESTful API calls (or equivalent internal function calls in Streamlit) to send the uploaded image to the backend (Module 6) and receive the bounding box data.

Learning Reflection Success: Achieved clean annotation visualization by correctly mapping the backend's bounding box coordinates (which are often normalized or based on the model's input size) to the actual displayed image dimensions in the UI.

Challenge/Takeaway: Ensuring no UI lag was a key challenge. This was solved by processing the image asynchronously after upload and optimizing the way the bounding box overlay was rendered (e.g., using a modern visualization library or efficient canvas operations).



Project Documentation

Module - 6

Project Title:

DermalScan: AI Facial Skin Aging Detection App

Project Statement:

The project aims to develop an AI-based system that can detect and classify facial skin aging signs such as wrinkles, dark spots, puffy eyes, and clear skin. It uses a pretrained EfficientNetB0 model along with computer vision techniques like Haar Cascades for face detection, preprocessing methods, and classification. A simple web interface allows users to upload images and view annotated outputs.

Project Statement

- Detect and highlight facial features related to aging.
- Classify into four categories: wrinkles, dark spots, puffy eyes, and clear skin.
- Achieve high accuracy through transfer learning with EfficientNetB0.
- Provide a user-friendly web interface for uploading and visualizing results.
- Build a backend pipeline for processing and inference.
- Enable exporting of results as annotated images and CSV files.

Module Implemented

Module Tasks & Deliverables Evaluation Goal
Module 6: Backend Pipeline for Model Inference Modularize code, load EfficientNet, log predictions/bounding boxes, and create an Integrated backend script.

Tasks & Deliverables Evaluation Goal
Module 6: Backend Pipeline for Model Inference Modularize code, load EfficientNet, log predictions/bounding boxes, and create an Integrated backend script. Seamless flow with \le 5 seconds per image inference time.

Technology Stack

Model: EfficientNet (Core machine learning model for inference).
Frameworks: TensorFlow/PyTorch (For model loading and execution) and a Python Web Framework (e.g., Flask/FastAPI) for the integrated backend API.
Libraries: OpenCV/PIL (For image preprocessing).
Integration: RESTful API (For end-to-end communication with the UI).
3. Learning Reflection

Learning Reflection

Success: Successfully modularized the inference code, which simplified testing and debugging.

Challenge/Takeaway: Meeting the \le 5 second per image latency requirement necessitated significant performance tuning around model loading and GPU/CPU resource management. The importance of efficient preprocessing on performance was a key learning point.