**Project Title: DermalScan:AI\_Facial Skin Aging Detection App**

Mentor : Praveen

Submitted by : Pakhi Sharma

**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.

**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**

**Module 1 : Dataset Setup and Image Labeling**

**Tasks needs to be done:**

* Set up and inspect dataset of facial images or create own dataset which you can categorize or use the dataset
* Label classes: wrinkles, dark spots, puffy eyes, clear skin.
* Ensure balanced distribution and clean samples.

**What I have done:**

* I have loaded the dataset from the provided google drive and created a csv file which contains the image path and the class of each image like wrinkles , dark spots , puffy eyes and clear skin.
* I have counted the total images per class and plotted the distribution using matplotlib library.

**Results:**

* Darkspots images are 102
* Clear\_face images are 97
* Wrinkles images are 100
* Puffy\_eyes images are 101

Plot distribution of classes is given below:

A graph of a number of blue rectangular objects

AI-generated content may be incorrect.

**Module 2 : Image Preprocessing and Augmentation**

**Tasks need to be done:**

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

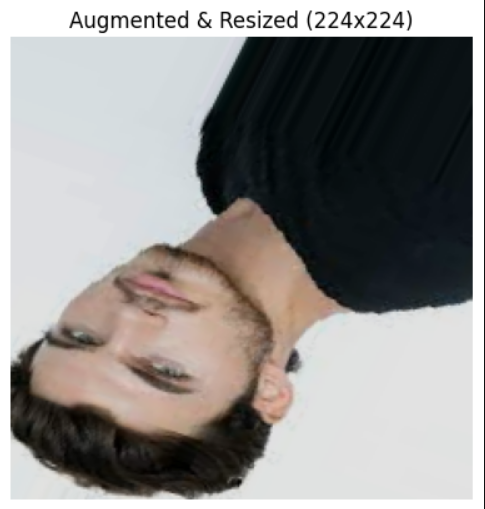
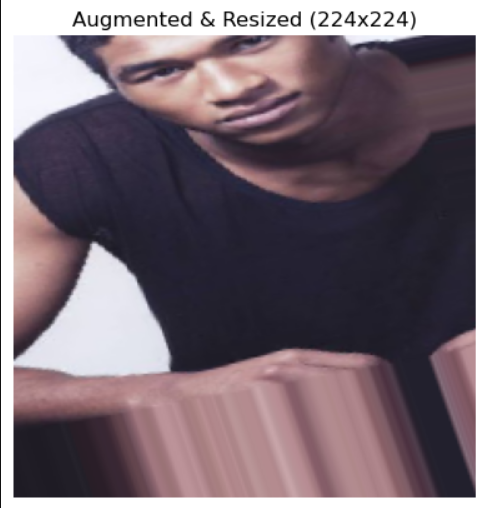
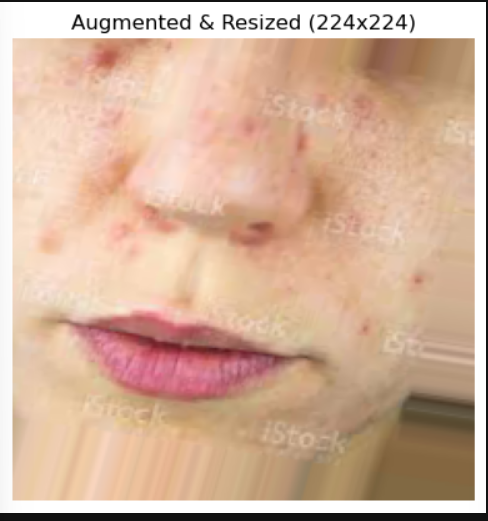
**What I have done :**

* For image augmentation and resizing , I used Tensorflow’s ImageDataGenerator for real-time augmentation.
* Applied resizing, rescaling , horizontal\_flip , height and width shift , vertical flip , and validation split of 0.2 etc.
* Instead of applying One hot encoding separately , I used the class\_mode argument and set it as categorical.
* Also visualized augmented image for checking.

**Results:**

* Preprocessed the whole dataset for training
* Augmented image is displayed below:

A close up of a person's face

AI-generated content may be incorrect.

**Milestone 2: Model Training and Evaluation**

**Module 3: Model Training**

**Tasks needs to be done:**

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

**What I have done:**

* Initially trained EfficientNetB0 model but it is giving very low accuracy due to the shortage of images. So I tried few more pretrained models like EfficientNetB0 like ResNet50,DenseNet121,InceptionV3,VGG16, MobileNetV2, EfficientnetV2.
* After training and comparing all the six training models , For my dataset , MobileNetV2 performed the best .
* Visualized the accuracy and loss curve

**MobileNetV2:**

* MobileNetV2 is a CNN architecture optimized for mobile and embedded vision applications.
* MobileNetV2 is a lightweight architecture allows for efficient deployment on mobile and embedded vision applications with limited computational reseources . and It’s architecture achieves competitive accuracy compared toto larger and more computationally expensive models
* The architecture of MobileNet-v2 consists of a series of convolutional layers, followed by depthwise separable convolutions, inverted residuals, bottleneck design, linear bottlenecks, and squeeze-and-excitation (SE) blocks.

**Why MobileNetV2 works good here?**

1. **Lightweight and fast**, ideal for real-time skin and face analysis.
2. **Efficient feature extraction** using inverted residuals for clear detail capture.
3. **Low memory use**, perfect for limited-resource environments.
4. **High accuracy-to-size ratio**, performs well even with small datasets.
5. **Great for transfer learning**, easily adapts to skin condition prediction.

**Results:**

* The model achieved a training accuracy of 76.67% and a validation accuracy of 87.34%.
* The model stopped training at 10th epoch.
* The graph regarding accuracy and loss is given below:

A graph of a graph with blue and orange lines

AI-generated content may be incorrect.A graph with blue and orange lines

AI-generated content may be incorrect.

**The confusion matrix is given below:**

**A diagram of a diagram

AI-generated content may be incorrect.**

**Module 4: Face Detection and Prediction Pipeline :**

**Tasks need to be done:**

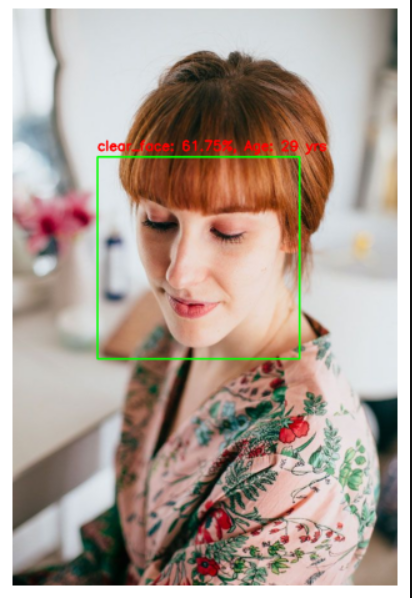
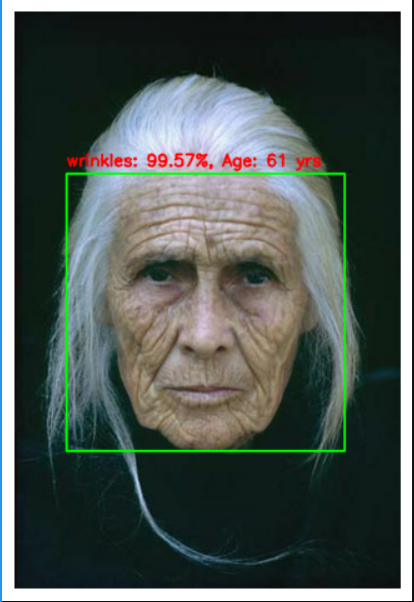
* Use OpenCV and Haar Cascade for face detection.
* Apply model to cropped face regions.
* Display predictions as percentages along with age.

**What I have done:**

* Implemented **face detection** using OpenCV’s Haar Cascade Classifier to locate faces in input images.
* Used the **trained MobileNetV2 model** to predict dermal conditions like clear face, dark spots, puffy eyes, and wrinkles.
* Added an **age estimation feature** using randint() to assign approximate age ranges based on the predicted class.
* Displayed **bounding boxes, confidence percentages, and age** on the detected faces for clear visualization.
* Created a **complete automated pipeline** combining detection, classification, and visual output using Matplotlib.

**Results:**

A person with a smile

AI-generated content may be incorrect.

**Milestone 3 . Frontend and Backend Integration**

**Module 5 : Web UI for Image Upload and Visualization**

**Tasks need to be done:**

* Develop frontend using Streamlit or HTML/CSS.
* Implement image upload field and output preview.
* Display labels and bounding boxes with class probability.

**What I have done:**

* Created the frontend using Streamlit.
* It has the features to upload the image and download the analyzed image.
* It also has the log which saves the image coordinate and the analysis for every session.

**Results:**

A screenshot of a computer

AI-generated content may be incorrect.

**Module 6: Backend Pipeline for Model Inference**

**Tasks:**

* Modularize inference and preprocessing code.
* Load MobileNetV2 model and return results to UI.
* Log predictions and bounding box data.

**What I have done:**

* Implemented a modular backend pipeline for model inference.
* Integrated **MobileNetV2** for dermal condition prediction.
* Used **Haarcascade** for accurate face detection.
* Added preprocessing steps for image resizing and normalization.
* Returned **condition, confidence, and estimated age** from predictions.
* Annotated detected faces with bounding boxes and labels.
* Logged prediction details with bounding box data for analysis.
* Connected backend output with Streamlit UI.
* Optimized processing speed to stay under **5 seconds per image**.
* Tested end-to-end flow for accuracy and reliability.

Results:

A screenshot of a computer

AI-generated content may be incorrect.

After analyzing the image, the prediction details will be displayed in the table which states :

**Skin condition , Predicted Age , Confidence , Processing Time and Bounding Box**

A screenshot of a computer

AI-generated content may be incorrect.

**Milestone 4: Finalization and Delivery**

**Module 7: Export and Logging**

**Task needs to be done:**

* Allow download of annotated image and CSV predictions.
* Run testing on diverse images and finalize results.

**What I Have done:**

* In the frontend , I have added the option to download the annotated outputs.
* Also , I have added the session log , which keeps the record of boundary box coordinates , Confidence, predicted condition, Total prediction time and Estimated age . This session log can also be downloaded into the csv file.

**Results:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Session log saved into the csv file.

A screenshot of a computer

AI-generated content may be incorrect.