Project Title: DermalScan: AI_Facial Skin Aging Detection App

Infosys SpringBoard Virtual Internship Program

Subimtted by:

Ashritha Ambati to Mr.Praveen sir

Problem Statement

Skin-related issues such as dark spots, wrinkles, and puffy eyes are common concerns affecting people of different ages. Accurate and early identification of these conditions can support better skincare decisions and help detect underlying health problems.

Manual diagnosis is often subjective, time-consuming, and requires expert knowledge, making automated computer-vision solutions highly beneficial.

To build an intelligent skin-analysis model, the first and most important requirement is to ensure that the image dataset is well-organized and properly represented. Without a correctly structured dataset, the performance of machine learning algorithms becomes unreliable. Additionally, skin-image datasets are often limited or imbalanced, which can negatively affect model accuracy — therefore, data augmentation is necessary to increase diversity and prevent bias.

Outcomes

- Collected and organized skin images into four categories
- Performed image validation and dataset statistics
- Applied data augmentation to improve dataset size and balance
- Preprocessed images for model training
- Built and evaluated a CNN model for skin-condition classification
- System can classify wrinkles, dark spots, puffy eyes, and clear skin automatically

Introduction

Skin-related conditions such as wrinkles, puffy eyes, and dark spots are common cosmetic and dermatological concerns. Early detection helps individuals make informed skincare decisions and can reflect underlying health conditions. Manual examination by dermatologists can be time-consuming, subjective, and inaccessible to many people. Artificial Intelligence, particularly Deep Learning, enables automated and accurate skin-condition assessment from facial images. This project focuses on identifying specific signs of facial skin aging using image classification techniques.

Milestone 1: Dataset Preparation and Preprocessing

Module 1: Dataset Setup and Image Labeling

• Collected the facial image dataset from the provided online source.

• Performed manual inspection to remove blurry, duplicate, or irrelevant images.

•Defined four classification categories: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.

• Labeled each image accurately based on visible facial features.

• Organized the dataset into separate folders corresponding to each class.

• Checked and ensured balanced class distribution to avoid training bias.

• Generated a class distribution plot for analysis and reporting.

Bar Chart Creation

```
plt.bar(classes, counts, color=colors)
```

- This is the core line that **draws the bar chart**.
- Each bar represents a class (classes) and its height is the number of images (counts).
- colors lets you customize the look of each bar.

```
plt.show()
```

- Finally, this command **renders the chart** so you can see it

Class Distribution Plot:

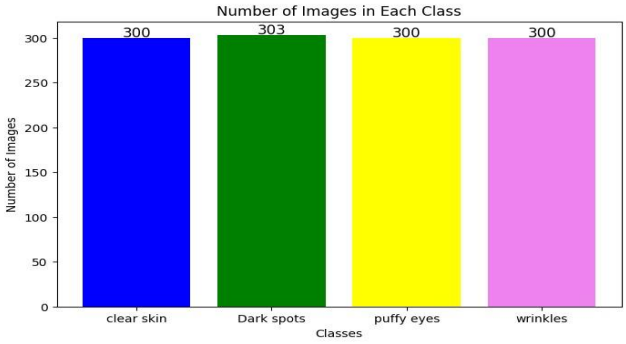

Fig1:Distribution of Images in Bar Graph

Module 2: Image Prepocessing and Augumentation

• Imported all images from the cleaned and labeled dataset prepared in Module 1.

• Performed image resizing to a fixed resolution of 224 × 224, ensuring consistent input size for the EfficientNetB0 model.

• Applied image normalization to scale pixel values between 0 and 1 for stable model training.

• Implemented image augmentation techniques such as: Horizontal flip, Rotation, Zoom, brightness etc.

• Generated an augmentation script to automate preprocessing and visualize sample augmented images.

• Converted categorical class labels into one-hot encoded vectors to match the model's training requirements.

• Verified augmentation quality by inspecting several augmented samples for realism and diversity.

• Ensured that the final preprocessed dataset maintains balanced class distribution after augmentation.

Augmentation Code

```
aug = ImageDataGenerator( rotation_range=30,
                          zoom_range=0.25,
                         width_shift_range=0.1,
                         height_shift_range=0.1,
                         brightness_range=[0.7, 1.3],
                         horizontal_flip=True,
                         vertical_flip=True )
```

ImageDataGenerator:
This creates an augmentation generator with specific transformations. Each parameter defines how the images will be randomly altered:

- rotation_range=30 → randomly rotate images up to 30 degrees.
- zoom_range=0.25 → randomly zoom in/out up to 25%.
- width_shift_range=0.1 → shift images horizontally by up to 10% of width.
- height_shift_range=0.1 → shift images vertically by up to 10% of height.
- brightness_range=[0.7, 1.3] → adjust brightness randomly between 70% (darker) and 130% (brighter).
- horizontal_flip=True → flip images left-to-right.
- vertical_flip=True → flip images top-to-bottom.

This configuration defines the **rules of augmentation**.

Augmented Images:

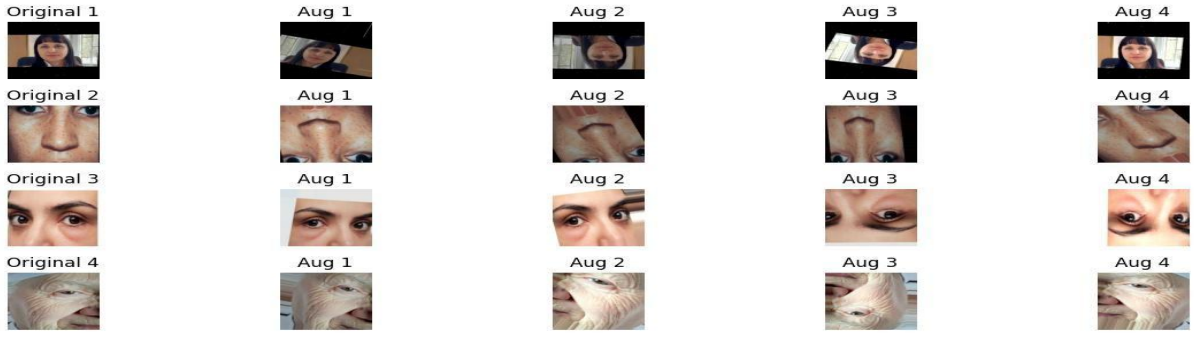

Fig2: visualization of Augmented Images

Libraries :

1. TensorFlow / Keras

Modules Used: load_img, img_to_array, ImageDataGenerator

Purpose: Loading input images into a workable format. Converting images to numerical arrays for processing. Applying augmentation techniques (rotation, zoom, shift, brightness, flip, shear), Managing preprocessing pipelines for deep-learning workflows

2. NumPy

Purpose: Numerical operations on image arrays, Array reshaping, normalization, and manipulation, Supporting backend computations for augmentation

3. Matplotlib (matplotlib.pyplot)

Purpose: Visualizing original images, Displaying augmented images in grid format. Validating dataset structure and augmentation results

4. OS Library (import os)

Purpose: Folder traversal , Dataset loading import os