Infosys SpringBoard Virtual Internship Program

DermalScan: AI Facial Skin Aging Detection App

Submitted by:
Suganth S S

Under the guidance of Mentor:
Mr. Praveen

Abstract

This project focuses on preparing a facial skin-aging dataset for an AI model that classifies conditions such as dark spots, wrinkles, puffy eyes, and clear skin. The images were organized, preprocessed, and enhanced using augmentation techniques to improve data quality and diversity. These steps provide a strong foundation for effective model training in the next module.

Table of Contents

---

1. Introduction

AI DermalScan is a deep learning-based system designed to classify facial aging features from images. The project leverages convolutional neural networks, image preprocessing, and face detection to build an end-to-end solution capable of analyzing facial skin conditions.

---

2. Problem Statement

---

3. Objectives

- Build a dataset categorized into facial aging features.
- Preprocess and augment images for optimal model performance.
- Train an EfficientNetB0-based classifier for aging sign detection.
- Implement face detection using Haar Cascades.
- Develop a user-friendly web interface for image upload and output visualization.
- Build backend integration for seamless inference.
- Allow exporting annotated images and logs.

---

4. Module-wise Implementation

Module 1: Dataset Setup & Image Labeling

In this module, the dataset required for the AI DermalScan model was prepared and analyzed. The following tasks were completed:

- Created the main dataset folder with four class categories:
  clear skin, **dark spots**, **puffy eyes**, **wrinkles**
- Verified that all images were correctly placed, readable, and properly labeled
- Counted the number of images in each class to understand dataset distribution
- Generated a simple table showing the image count per category
- Plotted a bar chart to visually compare the number of images across classes

These steps ensured that the dataset is structured and ready for preprocessing in Module 2.

Python Code Snippet for Image Count:

```
for cls in classes:
    count = len(os.listdir(os.path.join(dataset_path, cls)))
    data.append([cls, count])
```

Python Code Snippet for Dataset Visualization:

```
df.plot(kind="bar", x="Class", y="Image Count")

plt.title("Dataset Class Distribution")

plt.xlabel("Classes")

plt.ylabel("Number of Images")

plt.show()
```

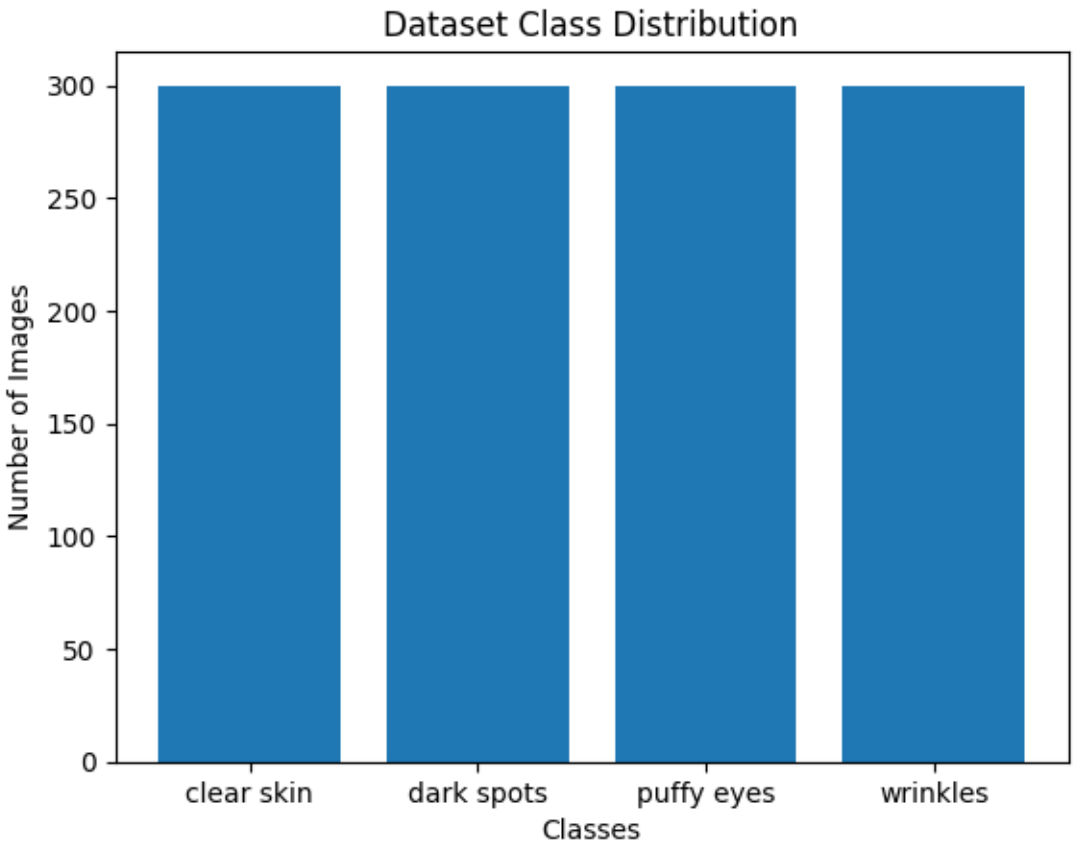

Fig 1 : Class-Wise Image Distribution in the Dataset

This bar chart illustrates the number of images available in each category: clear skin, dark spots, puffy eyes, and wrinkles. The distribution helps determine dataset balance and guides augmentation strategies for the following module.

Module 2: Image Preprocessing & Augmentation

In this module, the prepared dataset was preprocessed and augmented to improve model training quality.
The following tasks were completed:

- Resized all images to 224 × 224 to match model input requirements
- Normalized pixel values to the 0–1 range for efficient training
- Applied multiple augmentation techniques to enrich dataset variety, including:
    - Rotation
    - Horizontal flipping
    - Zoom transformation
    - Brightness variation
    - Shifting & Shearing
- Generated augmented samples to visually verify preprocessing quality

These steps ensured that the dataset becomes more diverse, reducing overfitting and improving model generalization in the next module.

Python Code Snippet Used for Augmentation:

```
datagen = ImageDataGenerator(

rescale=1.0 / 255.0,

rotation_range=25,

zoom_range=0.2,

horizontal_flip=True,

brightness_range=(0.6, 1.4)

)
```

Visualization of Augmentation Results:

This figure shows the original input image alongside four augmented versions generated using rotation, zoom, brightness adjustments, and spatial transformations. These augmentations help increase dataset diversity and improve model generalization during training.

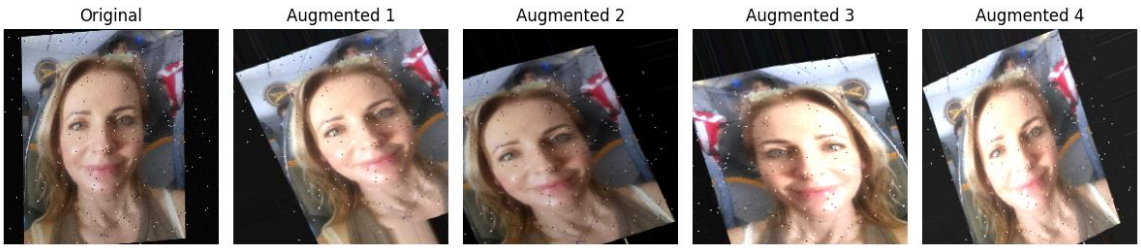

Fig 2: Original Image and Augmented Variants

---

5. Tools & Technologies

Languages:

- Python

Libraries:

- TensorFlow / Keras
- OS
- NumPy
- Pandas
- Matplotlib
- Pillow
- Scipy

Platforms:

- VS Code
- Jupyter Notebook
- GitHub

---

6. Conclusion
