Internship Project Report

DermalScan: AI Facial Skin Aging Detection App

Submitted by

Vishruth M Hullatti

Under the guidance of Mentor **Mr. Praveen**

Infosys Springboard Virtual Internship Program

Problem Statement

Develop an AI-powered web app using EfficientNetB0 that detects and classifies facial aging signs (wrinkles, dark spots, puffy eyes, clear skin) from user-uploaded images. The system localizes features with Haar Cascades, provides probability-based predictions with bounding boxes, and delivers annotated results in ≤5 seconds for consumer skincare applications.

Objectives

Develop a deep learning system using pretrained EfficientNetB0 to detect and classify facial aging signs (wrinkles, dark spots, puffy eyes, clear skin) from images with ≥90% accuracy.

Key Goals

- Localize aging features via Haar Cascades face detection and annotate with bounding boxes and probability percentages.
- Create a Streamlit web frontend for image upload, real-time inference, and result visualization in ≤5 seconds.
- Prepare balanced dataset, apply augmentation, train/evaluate model, and enable export of annotated images and CSV logs.

Milestone 1: Dataset Preparation and Preprocessing

Module 1: Dataset Setup and Image Labelling

The dataset was sourced from the provided Google Drive repository containing facial images categorized into four aging sign classes: wrinkles, dark spots, puffy eyes, and clear skin. Images were organized into class-specific subfolders under the dataset path.

1. Cleaned and labelled dataset:

```
if os.path.exists(class_path):
      file_count = len([f for f in os.listdir(class_path) if
      f.lower().endswith(('.jpg', '.jpeg', '.png'))])
      counts.append(file_count)
else:
      print(f"Warning: Folder '{cls}' not found at {class_path}")
      counts.append(0)
```

Validated folder structure and enumerated image files per class using os.listdir().

2. Class distribution plot

```
plt.figure(figsize=(10, 6))
bars = plt.bar(classes, counts, color=['#FF9999', '#66B2FF',
'#99FF99', '#FFCC99'])
```

Output:

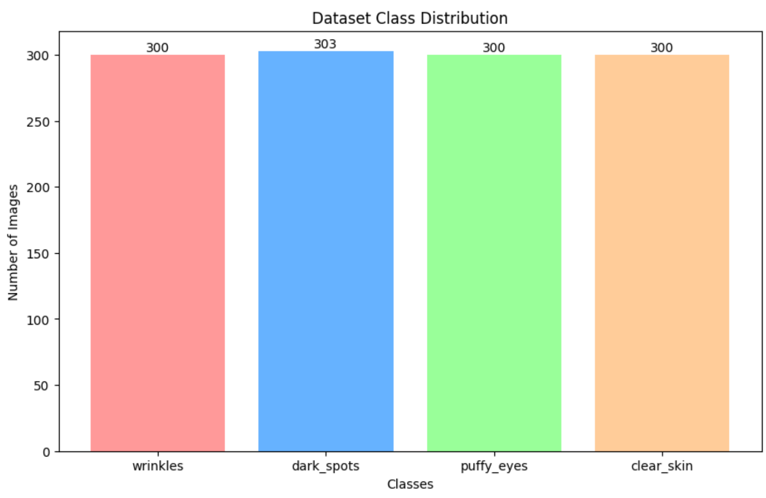

Figure 1: Class Distribution Bar Chart

Module 2: Image Preprocessing and Augmentation

Images were resized to 224x224 pixels and normalized using rescale=1./255 for EfficientNetB0 compatibility. Data augmentation applied rotation (30°), width/height shifts (0.2), zoom (0.2), and horizontal flips via ImageDataGenerator to enhance model robustness. An 80/20 train/validation split generated 963 training and 240 validation images with one-hot encoded labels. Batch visualization confirmed augmentation quality and class diversity retention.

1. Resizing and normalizing images

```
import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# speficifying image sizes

IMG_HEIGHT = 224

IMG_WIDTH = 224

BATCH_SIZE = 32
```

2. Applying image augmentation (flip, rotation, zoom)

```
train_datagen = ImageDataGenerator(

    rescale=1./255,                 # Normalize pixel values

    rotation_range=30,              # Rotation

    zoom_range=0.2,                 # Zoom

    horizontal_flip=True,           # Flip

)
```

3. Encoding class labels using one-hot encoding

```
train_generator = train_datagen.flow_from_directory(

class_mode='categorical',

)
```

Output:

```
Class Mappings: {'clear_skin': 0, 'dark_spots': 1, 'puffy_eyes': 2,
'wrinkles': 3}
```

4. Visualisation of Image Augmentation