DermalScan:AI_Facial Skin Aging Detection App

Infosys springboard virtual internship 6.0

Submitted By:

Gargi Kulkarni

Under the guidance of Mr. Praveen

Problem Statement:

There is currently no tool that can automatically evaluate the presence of age-related facial characteristics on a person's face without having to go to a dermatologist for an examination or wait until after an experimental study has been completed. Researchers have created a system, based upon deep learning, to provide users with real-time analysis of their facial images and receive immediate recommendations based upon these results.

Objectives:

- Detecting and locating facial characteristics which indicate the presence of age.
- Categorise the characteristics into 4 specific categories (i.e. Wrinkled, Dark Spot, Puffy Eyes and Clear Skin) using a Convolutional Neural Network (CNN).
- Create a user-friendly web based front end by allowing users to upload images with the expected annotated outcomes and percentage predictions.
- Connect the pipeline with a backend to process uploaded images, make inferences and return annotated results.

Technologies & Libraries Used:

- **Python (v3.10.1):** The primary language used for all data processing and model building.
- **Pandas(2.3.3):** Used for creating DataFrames, managing the dataset, and handling CSV files (import pandas as pd).
- **Matplotlib(3.10.7):** It is the foundational plotting and visualization library for Python.
- **Seaborn(0.13.2):** It simplifies complex statistical plots (like heatmaps or count plots) into single lines of code and makes them look better automatically.
- **Scikit-learn(1.7.2):** Used for splitting the data into training and testing sets (train_test_split).

Development Tools

- **Visual Studio Code (VS Code):** The Integrated Development Environment (IDE) used to write and run your code.
- **Jupyter Notebook:** Used for interactive coding and ensuring the code runs step by step.

Methodology:

Milestone 1: Dataset Preparation and Preprocessing-

➢ Module 1: Dataset Setup and Image Labeling

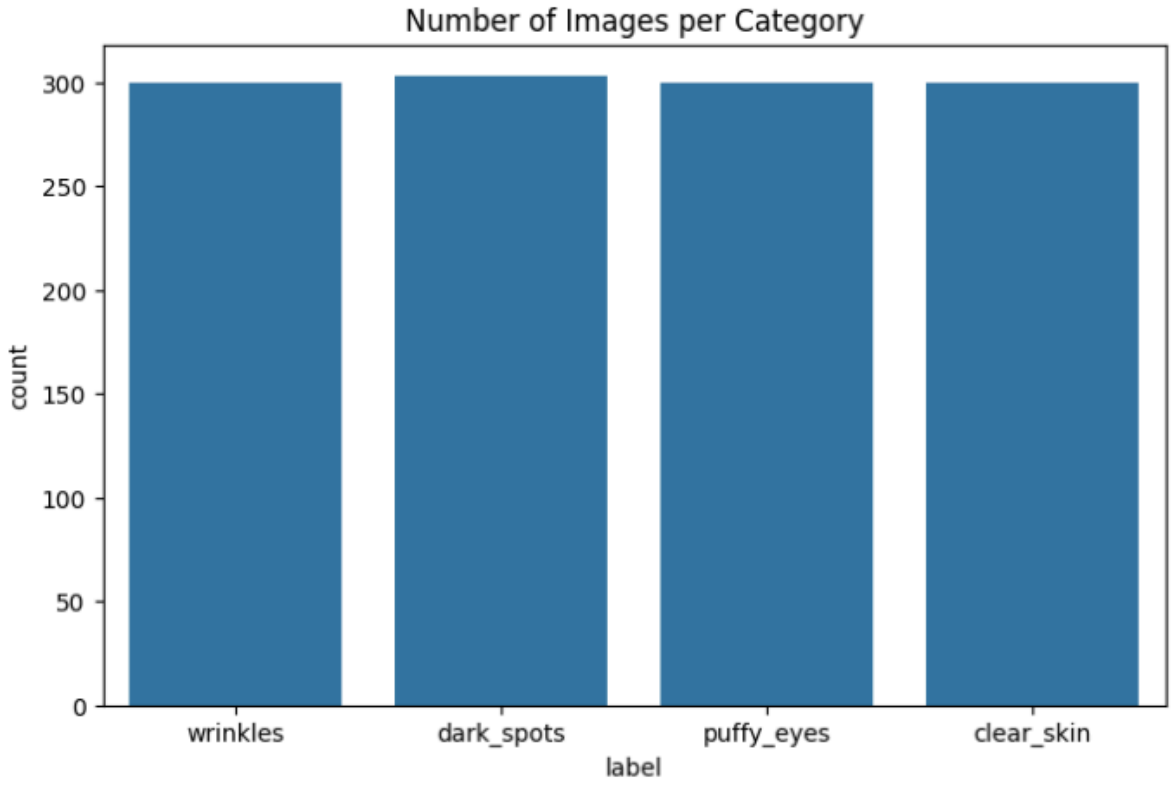
- Aggregated facial images into four distinct classes: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.
- Manually filtered out poor-quality samples and ensured equal representation across all classes to prevent model bias.
- Produced a finalized, labeled dataset directory and a class distribution plot confirming data balance.

➢ Code:

```
import os
import pandas as pd
base_dir = "."
categories = ["wrinkles", "dark_spots", "puffy_eyes", "clear_skin"]
data = []

print("Starting to scan folders...")

for label in categories:
folder = os.path.join(base_dir, label)
if os.path.exists(folder):
files = [f for f in os.listdir(folder) if f.lower().endswith((".jpg", ".png", ".jpeg"))]
print(f"Found {len(files)} images in: {label}")

 for file in files:
      data.append([file, label])
   else:
      print(f"ERROR: Could not find folder: {folder}")

# Create the table
df = pd.DataFrame(data, columns=["filename", "label"])
print("-" * 30)
print(f"Total images loaded: {len(df)}")

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 5))
sns.countplot(x='label', data=df)
plt.title('Number of Images per Category')
plt.show()
```

➢ **Output:**

```
Starting to scan folders...
Found 300 images in: wrinkles
Found 303 images in: dark_spots
Found 300 images in: puffy_eyes
Found 300 images in: clear_skin
-----------------------------
Total images loaded: 1203
```



➢ Module 2: Image Preprocessing and Augmentation

- We have successfully transformed raw images into a clean format ready for model training.
- Organized the facial images into four distinct classes: Wrinkles, Dark Spots, Puffy Eyes, and Clear Skin.
- Resized all images to 224x224 pixels and normalized pixel values so the model receives consistent input.
- Applied transformations like rotation, flips, and zoom to artificially increase the dataset size and help the model generalize better.

➢ Code:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import os
        IMG_HEIGHT = 224
        IMG_WIDTH = 224
        BATCH_SIZE = 32
        DATA_DIR = "." # <--- REPLACE THIS with your folder path
train_datagen = ImageDataGenerator(
   rescale=1./255,            # Normalization: Scales pixel values from 0-255 to 0-1
   rotation_range=20,         # Augmentation: Random rotation
   zoom_range=0.2,            # Augmentation: Random zoom
   horizontal_flip=True,      # Augmentation: Random horizontal flip
   width_shift_range=0.1,     # Optional: Shift width slightly
   height_shift_range=0.1,    # Optional: Shift height slightly
   fill_mode='nearest',       # Strategy for filling in newly created pixels
   validation_split=0.2       # reserving 20% for validation (optional)
)

train_generator = train_datagen.flow_from_directory(
   DATA_DIR,
   target_size=(IMG_HEIGHT, IMG_WIDTH), # Resize to 224x224
   batch_size=BATCH_SIZE,
   class_mode='categorical',          # THIS performs One-Hot Encoding
   subset='training',
   shuffle=True
)

def visualize_augmentation(generator):
            x_batch, y_batch = next(generator)
class_dict = generator.class_indices
label_map = {v: k for k, v in class_dict.items()}
        plt.figure(figsize=(12, 12))
        plt.suptitle("Preprocessed & Augmented Images (224x224)", fontsize=16)
```

```
    for i in range(9): # Show first 9 images
        plt.subplot(3, 3, i + 1)

        # Display image
        plt.imshow(x_batch[i])
        class_index = np.argmax(y_batch[i])
        class_name = label_map[class_index]

        plt.title(f"Label: {class_name}\nShape: {x_batch[i].shape}")
        plt.axis('off')
        plt.tight_layout()
        plt.show()
try:
    visualize_augmentation(train_generator)
except Exception as e:
     print(f"Error: Could not visualize. Make sure 'DATA_DIR' points to a folder containing class subfolders. \nError details: {e}")
```

➢ Output