DermalScan :AI_Facial Skin Aging Detection App

Infosys Springboard Virtual Internship Program

Submitted by

Parth Agrawal

Under the Guidance of

Mr. Praveen

Mentor, Infosys Springboard

1. Introduction

The goal of Milestone 1 is to complete all dataset-related tasks required for the AI Facial Skin Aging Detection App.
The Milestone 1 includes two modules:

- **Module 1:** Dataset Setup and Image Labelling.
- **Module 2:** Image Preprocessing and Augmentation.

This documentation describes the dataset preparation, cleaning, class distribution visualization, augmentation pipeline, and one-hot encoding implemented in your notebook.

2. Module 1: Dataset Setup and Image Labelling

2.1 Dataset Folder Structure

The dataset was organized in a directory structure where each folder represents a class label:

dataset/

├── wrinkles/

├── dark spots/

├── puffy eyes/

├── clear skin/

2.2 Dataset Scanning & Image Counting

A custom script was written to scan the dataset and count the number of images in each class:

```
def scan_dataset(dataset_dir=DATASET_DIR):
    class_counts = {}
    for cls in CLASSES:
        folder = os.path.join(dataset_dir, cls)
        count = len(os.listdir(folder))
        class_counts[cls] = count
    return class_counts
```

2.3 Duplicate Image Detection & Removal

To produce a clean dataset, a hash-based duplicate detection algorithm was implemented:

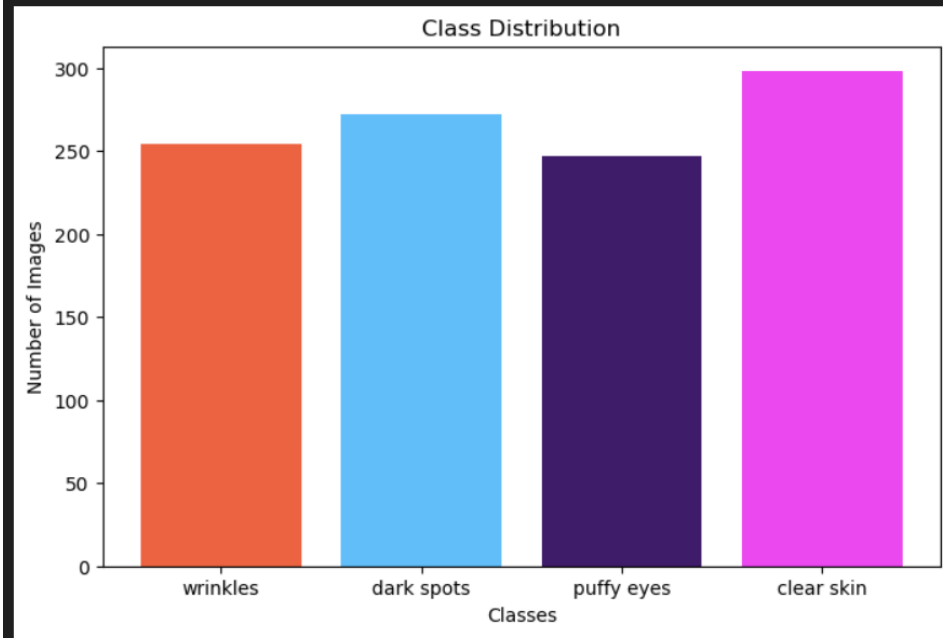
```
with open(  f:
  h = hashlib.md5(f.read()).hexdigest()
```

- If the hash already exists → image is a duplicate
- All duplicates were logged and removed automatically
- Dataset was re-scanned after cleanup

2.4 Class Distribution Visualization

```
def plot_class_distribution(counts):
 plt.figure(figsize=(8,5))
 colors = ["#FF5733", "#33C1FF", "#44196D", "#FF33F8"]
 plt.bar(counts.keys(), counts.values(),color=colors)
 plt.title("Class Distribution")
 plt.xlabel("Classes")
 plt.ylabel("Number of Images")
 plt.show()
```

```
Class Counts: {'wrinkles': 254, 'dark spots': 272, 'puffy eyes': 247, 'clear skin': 298}
```



3. Module 2: Image Preprocessing & Augmentation

As required in the project specification, Module 2 includes:

- resizing to 224×224
- normalization
- augmentation
- one-hot encoding

All four requirements were fully implemented.

3.1 Image Preprocessing with ImageDataGenerator

```
IMG_SIZE = (224, 224)
augmentation = ImageDataGenerator(
   rescale=1/255,
   rotation_range=10,
   zoom_range=0.1,
   horizontal_flip=True,
   width_shift_range=0.05,
   height_shift_range=0.05,
   validation_split=0.2
```

Preprocessing Achieved:

Requirement	Status
Resize to 224x224	Implemented
Normalize images	Using rescale=1/255
Augmentation (flip, zoom, rotation)	Implemented
Train/Validation split	80/20

3.2 Loading Data with One-Hot Encoding

```
train_gen = augmentation.flow_from_directory(
   "dataset",
   target_size=IMG_SIZE,
   batch_size=32,
   class_mode="categorical",
   subset="training"
)
```

This automatically converts each label into a **one-hot encoded vector.**

3.3 Augmentation Visualization

A 4×4 grid of augmented images was displayed:

```
for i in range(16):
   plt.subplot(4,4,i+1)
   plt.imshow(images[i])
   plt.axis("off")
```

This confirms the applied augmentations visually.

Conclusion:

Milestone 1 was successfully completed by preparing a clean and well-structured dataset, implementing preprocessing techniques, applying image augmentation, and enabling one-hot encoding for class labels. The dataset is now fully ready for model development in Milestone 2, where EfficientNetB0 will be trained using the processed inputs.