DermalScan : AI_Facial Skin Aging Detection App

Infosys Springboard Virtual Internship Program

Submitted by,

Rounak Kumar Mishra

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

Problem Statement

Skin-related issues such as wrinkles, dark spots, puffy eyes, and uneven skin texture are common in dermatology. Identifying these conditions manually is time-consuming and requires expert knowledge.
The **AI DermaScan** project aims to develop an automated AI system that can classify facial skin images into distinct categories such as Clear Skin, Dark Spots, Puffy Eyes, Wrinkles.To achieve accurate classification, the dataset must undergo **cleaning, preprocessing, augmentation, and normalization**.

Objectives

The main objectives of Milestone 1 is to inspect and validate the dataset, to resize all images to a standardized input size, to normalize image data for better model performance, to apply augmentation techniques (flip, rotation, zoom), to generate a preprocessed and balanced dataset, to visualize before-and-after augmentation results, to prepare the data for future modules (train/test split & model training)

Technologies & Libraries Used

- **Programming Language:** Python 3.11.9
- **Libraries:**

Library	**Purpose**
OpenCV (cv2)	Image loading, resizing, augmentation
NumPy	Numerical operations and normalization
Pillow (PIL)	Image verification and preview
Matplotlib	Visualization of samples and augmentation
Pandas	Data analysis and reporting

Development Tools

- Jupyter Notebook – primary development environment
- Windows OS
- Python Virtual Environment (myjupyterenv)

Methodology

The project followed a structured methodology:

Module 1: Dataset Setup and Image Labeling

Deliverables:

- Cleaned and labeled dataset
- Class distribution plot

1. Ensure balanced distribution and cleaned dataset.

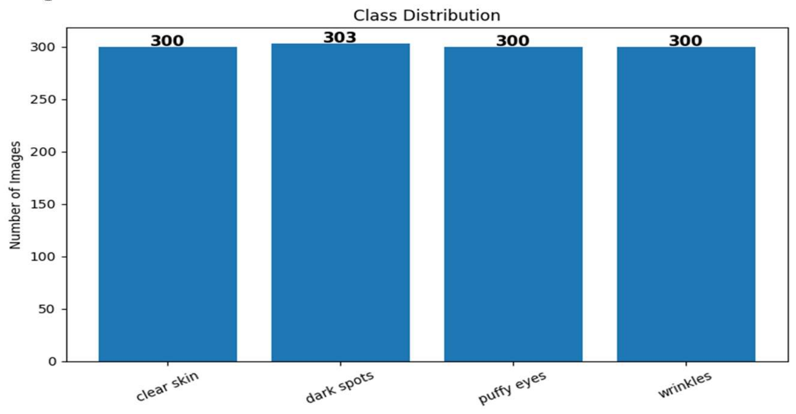
```
from PIL import Image
import os
root = "DATASET"
bad = []
for cls in os.listdir(root):
    cls_dir = os.path.join(root, cls)
    if not os.path.isdir(cls_dir):
        continue
    for f in os.listdir(cls_dir):
        p = os.path.join(cls_dir, f)
        try:
            with Image.open(p) as im:
                im.verify()
        except:
            bad.append(p)
print("Corrupted files:", len(bad))
for p in bad:
    print(p)
```

Output:
```
Corrupted files: 0
```

2. Class distribution plot

```
root = "DATASET"
counts = {}
for cls in os.listdir(root):
    cls_path = os.path.join(root, cls)
    if os.path.isdir(cls_path):
        counts[cls] = len(os.listdir(cls_path))
classes = list(counts.keys())
vals = [counts[c] for c in classes]
plt.figure(figsize=(8,5))
bars = plt.bar(classes, vals)
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2,height + 1,str(height),
      ha='center', fontsize=12, fontweight='bold')
plt.title("Class Distribution")
plt.ylabel("Number of Images")
plt.xticks(rotation=25)
plt.tight_layout()
plt.show()
```

Output:



Module 2: Image Preprocessing and Augmentation

Deliverables:

- Preprocessed and augmented dataset
- Augmentation script with visualization

1. Resize and normalize images (224x224).

```
TARGET_SIZE = (224, 224)  # (width, height)
def resize_image_save(src_path, dst_path, size=TARGET_SIZE):
    img = cv2.imread(src_path)
    if img is None:
        return False
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, size, interpolation=cv2.INTER_AREA)
    cv2.imwrite(dst_path, cv2.cvtColor(img, cv2.COLOR_RGB2BGR))
    return True
errors = []
for cls in classes:
    src_dir = os.path.join(ROOT, cls)
    dst_dir = os.path.join(PROCESSED, cls)
    os.makedirs(dst_dir, exist_ok=True)
    for fname in os.listdir(src_dir):
        src_p = os.path.join(src_dir, fname)
        dst_p = os.path.join(dst_dir, fname)
        if not resize_image_save(src_p, dst_p):
            errors.append(src_p)
print("Resize complete. unreadable files:", len(errors))
```

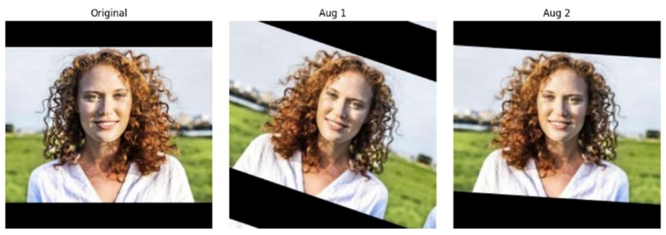
Output:
```
Resize complete. unreadable files: 0
```

2. Apply image augmentation & visualization.

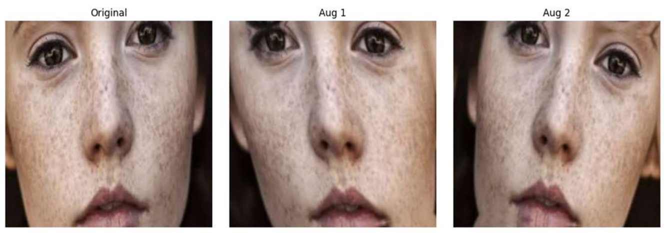
```
from PIL import Image
def      show_before_after_grid(processed_folder,      augmented_folder,
samples_per_class=2):
    classes = sorted(os.listdir(processed_folder))
    for cls in classes:
        print(f"\n=== CLASS: {cls} ===")
        proc_dir = os.path.join(processed_folder, cls)
        aug_dir  = os.path.join(augmented_folder, cls)
        proc_files = sorted([f for f in os.listdir(proc_dir)])
        aug_files  = sorted([f for f in os.listdir(aug_dir)])
        selected    =   random.sample(proc_files,   min(samples_per_class,
len(proc_files)))
        for f in selected:
            base = os.path.splitext(f)[0]
            orig_path = os.path.join(proc_dir, f)
            orig_img = Image.open(orig_path)
            related_aug = [a for a in aug_files if a.startswith(base) and
"aug" in a]
            total_cols = 1 + len(related_aug)
            plt.figure(figsize=(4 * total_cols, 4))
            plt.subplot(1, total_cols, 1)
            plt.imshow(orig_img)
            plt.title("Original")
            plt.axis("off")
            for idx, a in enumerate(related_aug, start=2):
                aug_img = Image.open(os.path.join(aug_dir, a))
                plt.subplot(1, total_cols, idx)
                plt.imshow(aug_img)
                plt.title(f"Aug {idx-1}")
                plt.axis("off")
            plt.tight_layout()
            plt.show()
show_before_after_grid("processed","augmented",samples_per_class=1)
```

Output:

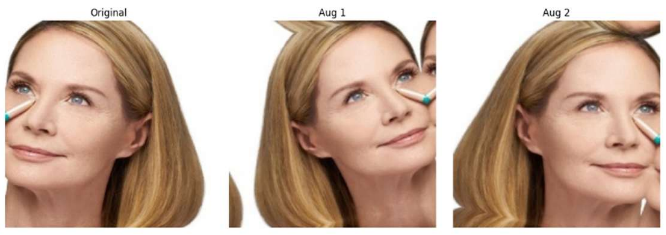
```
=== CLASS: clear skin ===
```



```
=== CLASS: dark spots ===
```



```
=== CLASS: puffy eyes ===
```



```
=== CLASS: wrinkles ===
```

3. Encode class labels using one-hot encoding.

```
mapping = {cls: idx for idx, cls in enumerate(classes)}
pd.DataFrame(list(mapping.items()),                          columns=["class",
"index"]).to_csv(os.path.join(OUTPUTS, "class_mapping.csv"), index=False)
print("Mapping saved:", mapping)
def to_one_hot(index, num_classes=len(classes)):
    arr = np.zeros(num_classes, dtype=int)
    arr[index] = 1
    return arr
```

Output:

```
Mapping saved: {'clear skin': 0, 'dark spots': 1, 'puffy eyes': 2,
'wrinkles': 3}
```