DermalScan:AI_Facial Skin Aging Detection App

Infosys SpringBoard Virtual Internship Program

Submitted By,

Priya Ghosal

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

Project Statement:

Facial skin aging shows signs like wrinkles, dark spots, and puffiness, which can be hard to identify correctly without a specialist. As AI is becoming more common in healthcare, there is a need for an easy, reliable, and automatic system to detect and classify these aging signs. This project aims to create such a system using deep learning and computer vision.

Expected outcome:

1. **A fully structured facial skin aging dataset** organized into four classes:

- Clear Skin
- Dark Spots
- Puffy Eyes
- Wrinkles

2. **A DataFrame (df)** containing all image filepaths and corresponding labels extracted from folder structure.
3. **Successfully preprocessed images**, resized to 256×256, normalized (0–1 scaling), and stored in an array ready for deep learning models.
4. **Automated data augmentation pipeline** using Keras ImageDataGenerator to improve dataset variability and reduce overfitting.
5. **Label encoding with one-hot vectors**, enabling compatibility with classification models.
6. **Class distribution visualization** for dataset balance analysis and **Augmentation preview visualization** showing how transformations modify sample images.

Tech Stack Used

1. Programming Language

- **Python 3.10.11**

2. Deep Learning Frameworks

- **TensorFlow / Keras**
    - Image loading and augmentation
    - One-hot encoding
    - Data pipeline utilities

3. Image Processing Libraries

- **OpenCV (cv2)**
  - Image reading
  - Resizing
  - Color conversion
- **Pillow (PIL)**
  - Safe image opening
  - Handling corrupted images

5. Visualization Libraries

- **Matplotlib**
  - Augmentation preview
  - Image display
- **Seaborn**
  - Class distribution plots

6. Machine Learning Tools

- **Scikit-Learn**
  - Train/Validation split
  - Data shuffling

7. Development Environment

- **Jupyter Notebook / JupyterLab**

MILESTONE 1: Dataset Preparation and Preprocessing

MODULE 1: Dataset Setup & Image Labeling

Module 1 focuses on preparing the dataset so the model can understand it later. In deep learning, data preparation is one of the most important steps because the model's accuracy completely depends on clean, well-organized data.

1. **Dataset Setup**
   The dataset was structured into four separate class folders — **Wrinkles**, **Dark Spots**, **Puffy Eyes**, and **Clear Skin** — each containing images representing that skin condition.

   ### Code Snippet

   ```
   base_dir = "data"
   classes = ["Wrinkles", "Dark Spots", "Puffy Eyes", "Clear Skin"]
   ```

   Folder structure is like:

   ```
   data/
   ├── Wrinkles
   ├── Dark Spots
   ├── Puffy Eyes
   └── Clear Skin
   ```

2. **Building a DataFrame of Filepaths & Labels**

   We create a function to walk through each class folder, pick up image paths, and store them with their correct label.

   This helps convert the raw dataset into a *machine-readable table*.

3. **Creating the Final Dataset Table**

   After collecting the image filepaths and class labels from all four folders, we organized this information into a structured pandas DataFrame. This table acts as the central dataset index, where each row represents one image along with its corresponding label. The DataFrame contains two main columns:

- filepath: the complete directory path to the image file

- label: the class to which the image belongs (Wrinkles, Dark Spots, Puffy Eyes, or Clear Skin)

Code Snippet

```
df = build_df_from_folders(base_dir, classes)

print("Total images:", len(df))
print(df["label"].value_counts())
df.head()
```

Output

label	**count**
Dark Spots	**303**
Wrinkles	**300**
Puffy Eyes	**300**
Clear Skin	**300**

4. Class Distribution

The bar chart shows the number of images available for each skin condition category in the dataset. There are four classes:

- Wrinkles
- Dark Spots
- Puffy Eyes
- Clear Skin

Code Snippet

```
plt.figure(figsize=(8,5))

colors = ["blue", "yellow", "green", "purple"]

sns.countplot(data=df, x="label", hue="label", palette=colors, legend=False)

plt.title("Class Distribution", fontsize=14)

plt.xlabel("Class")

plt.ylabel("Number of Images")

plt.xticks(rotation=20)

plt.tight_layout()

plt.show()
```

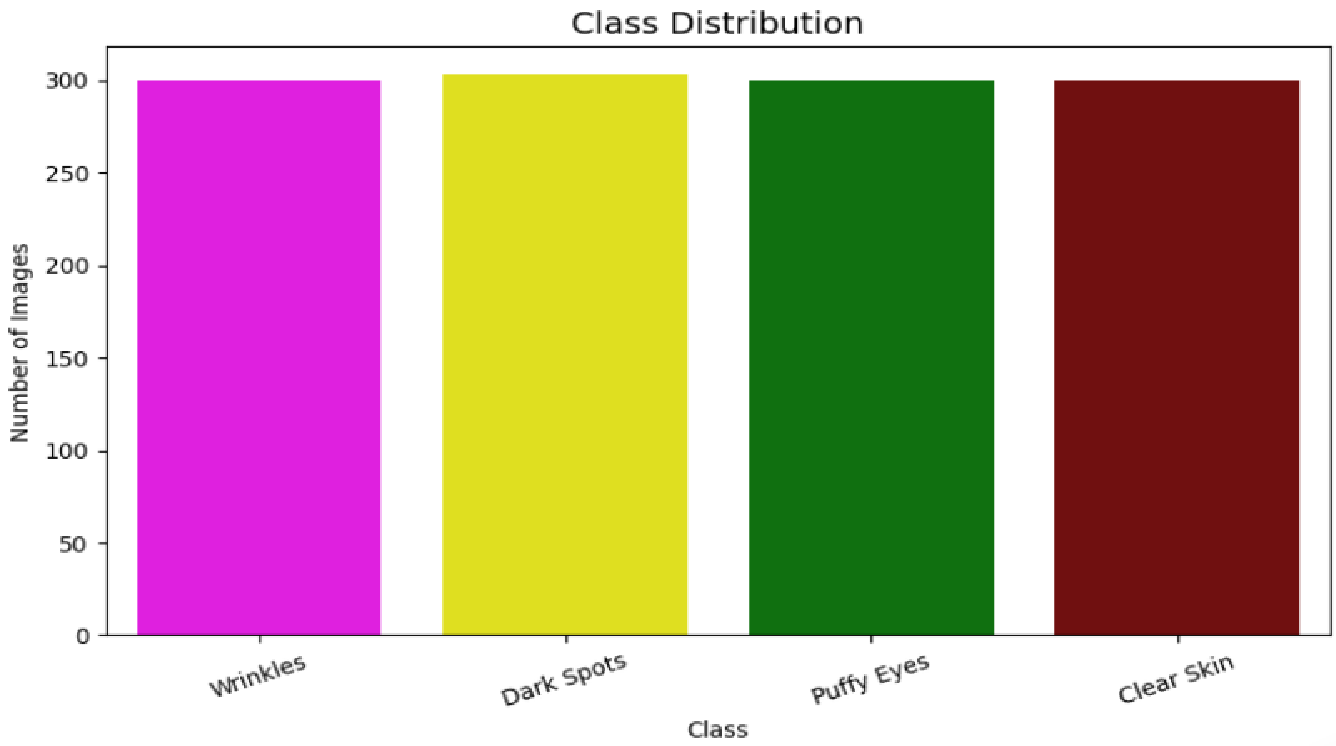

Fig 1. Class Distribution Plot

Observation From Fig. 1

- All classes contain **approximately 300 images** each.
- No class shows significant under-representation or over-representation.
- The distribution is nearly uniform across all four categories.

Importance of Balanced Distribution

- Prevents the model from becoming biased toward a class with more samples.
- Ensures fair learning and equal exposure to all facial aging conditions.
- Reduces the need for additional techniques like oversampling, undersampling, or class weighting.

Impact on Model Training

- Leads to **more stable training performance**.
- Helps improve model **accuracy, generalization, and robustness**.
- Provides a reliable foundation for preprocessing, augmentation, and later model training.

Module 2: Image Preprocessing and Augmentation

In this module, all facial images are resized to a fixed resolution of **224×224 pixels**, normalized to the range **[0, 1]**, and passed through a set of augmentation operations. These augmentations—such as rotation, flipping, shifting, and zooming—help artificially expand the dataset and introduce variability. This makes the model more generalizable and reduces overfitting.

Additionally, class labels are encoded using **one-hot encoding**, which converts categorical labels (Clear Skin, Dark Spots, Puffy Eyes, Wrinkles) into numerical vectors suitable for training a multi-class classification model.

1. Image Loading & Preprocessing

Every image from Module 1 (df) is:

- Loaded with Pillow (PIL)
- Converted to RGB (ensures color consistency)
- Resized to **224 × 224** (IMG_SIZE = 224)
- Normalized to values between **0 → 1**

2. Data Augmentation

We use **ImageDataGenerator** to apply transformations such as:

- Rotation
- Horizontal shift
- Vertical shift
- Zoom
- Horizontal flipping

Code Snippet:

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.15,
    horizontal_flip=True,
    fill_mode='nearest')
```

Fig 2. Augmented Image Preview

3. Label Encoding

```
label_map = {
    "Clear Skin": 0,
    "Dark Spots": 1,
    "Puffy Eyes": 2,
    "Wrinkles": 3
}
```

Then labels are converted to **one-hot vectors**, such as:

```
Wrinkles → [0, 0, 0, 1]
```

5. Train–Validation Split

This allows testing the model on unseen data to measure performance.

Code Snippets:

```
X_train, X_val, y_train, y_val = train_test_split(
    images,
    labels_onehot,
    test_size=0.2,
    random_state=42,
    shuffle=True
)
```

Conclusion

Milestone 1 successfully prepares the dataset through structured organization, preprocessing, labeling, and augmentation. This ensures the data is clean, balanced, and sufficiently diverse to support high-performance model training in subsequent modules. Both Module 1 and Module 2 meet the milestone requirements and complete the foundation for the **DermalScan AI Facial Aging Detection System**.