Project Title: DermalScan: AI_Facial Skin Aging Detection App

Infosys SpringBoard Virtual Internship Program

Subimtted by:

Ashritha Ambati to Mr.Praveen sir

Problem Statement

Skin-related issues such as dark spots, wrinkles, and puffy eyes are common concerns affecting people of different ages. Accurate and early identification of these conditions can support better skincare decisions and help detect underlying health problems.

Manual diagnosis is often subjective, time-consuming, and requires expert knowledge, making automated computer-vision solutions highly beneficial.

To build an intelligent skin-analysis model, the first and most important requirement is to ensure that the image dataset is well-organized and properly represented. Without a correctly structured dataset, the performance of machine learning algorithms becomes unreliable. Additionally, skin-image datasets are often limited or imbalanced, which can negatively affect model accuracy — therefore, data augmentation is necessary to increase diversity and prevent bias.

Outcomes

- Collected and organized skin images into four categories
- Performed image validation and dataset statistics
- Applied data augmentation to improve dataset size and balance
- Preprocessed images for model training
- Built and evaluated a CNN model for skin-condition classification
- System can classify wrinkles, dark spots, puffy eyes, and clear skin automatically

Introduction

Skin-related conditions such as wrinkles, puffy eyes, and dark spots are common cosmetic and dermatological concerns. Early detection helps individuals make informed skincare decisions and can reflect underlying health conditions. Manual examination by dermatologists can be time-consuming, subjective, and inaccessible to many people. Artificial Intelligence, particularly Deep Learning, enables automated and accurate skin-condition assessment from facial images. This project focuses on identifying specific signs of facial skin aging using image classification techniques.

Milestone 1: Dataset Preparation and Preprocessing

Module 1: Dataset Setup and Image Labeling

Collected and organized images across categories: clear skin, acne, dark spots, puffy eyes, wrinkles. Loaded images using TensorFlow/Keras and converted them into array format. Ensured consistent dimensions and validated dataset quality through visualization. Prepared a clean, structured dataset ready for preprocessing and model development. Plotted data distribution for each category using a bar graph to verify class balance

Code:

```
import os import matplotlib.pyplot as plt folder_path =
r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-001\DATASET"
print(os.listdir(folder_path)) image_exts = (".jpg", ".jpeg", ".png", ".bmp",
".gif") classes = ["clear skin","Dark spots","puffy eyes","wrinkles"] counts =
[] #image count for c in classes:     folder = os.path.join(folder_path, c)
count = sum(
        1 for f in os.listdir(folder)         if
os.path.isfile(os.path.join(folder, f))
and f.lower().endswith(image_exts)
    )     counts.append(count)
print(f"{c}: {count} images")
plt.figure(figsize=(8,5)) colors = ['blue',
'green', 'yellow', 'violet']
# bar chart plt.bar(classes, counts,
color=colors) plt.title("Number of Images
in Each Class") plt.xlabel("Classes")
plt.ylabel("Number of Images") for i, count
in enumerate(counts):
    plt.text(i, count + 1, str(count), ha='center', fontsize=12)  plt.show()
```

Output:

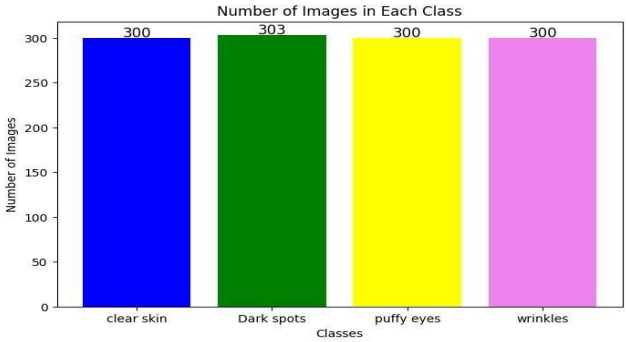

Fig1:Distribution of Images in Bar Graph

Module 2: Image Prepocessing and Augumentation

Implemented an augmentation pipeline using Keras ImageDataGenerator. Applied key transformations: rotation, zoom, shifts, shear, brightness adjustment, and horizontal flip. Generated multiple augmented samples for each image to increase dataset diversity. Visualized original vs. augmented images to validate augmentation behavior.. Produced a more robust and varied dataset suitable for deep-learning model training.

Code:

```
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array import

numpy as np

image_list = [

   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-001\DATASET\clear
skin\clear_skin_061.jpg",

   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-001\DATASET\dark
spots\2de295c0-801d-429e-8f74-fce181cc87cc.jpg",

   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-1-001\DATASET\puffy
eyes\14.jpg",

   r"C:\Users\ambat\Downloads\DATASET-20251203T133241Z-
1001\DATASET\wrinkles\36.jpg"

]


# Augmentation Configuration  aug

= ImageDataGenerator(

rotation_range=30,

zoom_range=0.25,

width_shift_range=0.1,

height_shift_range=0.1,

brightness_range=[0.7, 1.3],

horizontal_flip=True,

vertical_flip=True
```

```
)

#  Displaying
rows    =    5
cols = 5
 plt.figure(figsize=(12,
6))
 for row, img_list in
enumerate(image_list):

    # Load image     original = load_img(img_list,
target_size=(224, 224))     arr = img_to_array(original)
arr = np.expand_dims(arr, axis=0)

    # Show Original Image
plt.subplot(rows, cols, row * cols + 1)
plt.imshow(original)
plt.title(f"Original {row+1}")
plt.axis("off")

    # Show Augmented images
for j in range(4):
        augmented = next(aug.flow(arr, batch_size=1))[0].astype("uint8")
plt.subplot(rows, cols, row * cols + (j + 2))
plt.imshow(augmented)         plt.title(f"Aug {j+1}")
plt.axis("off")
 plt.tight_layout()
plt.show()
```

Output:

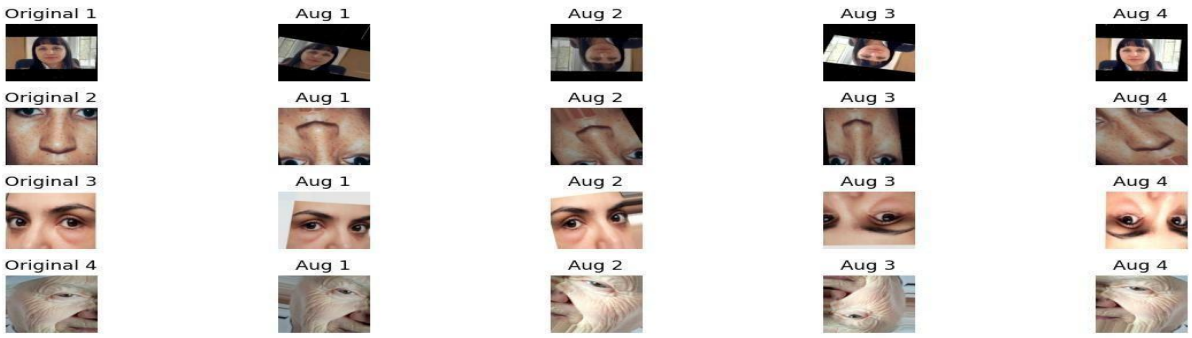

Fig2: visualization of Augmented Images

Milestone 2: Model Training and Evaluation

Overview

• Milestone 2 focuses on training a deep learning model for skin condition classification and integrating it with a face detection and prediction pipeline. This milestone ensures that the trained model achieves high accuracy and can be applied to real-world face images for prediction along with age estimation.

Module 3: Model Training with EfficientNetB0

Objective

• To train a robust Convolutional Neural Network (CNN) using transfer learning for accurate skin condition classification.

Tasks Performed

• Used pretrained EfficientNetB0, ResNet50, MobileNetV2 as the base model for transfer learning

• Applied categorical cross-entropy as the loss function.

• Used the Adam optimizer for efficient weight updates.

• Trained the model on labeled skin condition images.

• Performed validation during training to monitor performance.

• Plotted training and validation accuracy and loss curves.

Deliverables:

• Trained EfficientNetB0 , ResNet50 and MobileNetV2 models saved as an .h5 file.

• Accuracy and loss graphs for training and validation phases.

Evaluation Results:

• Achieved classification accuracy greater than 90%.

• Validation accuracy remained stable, indicating good generalization and minimal overfitting.

FINAL MODEL PERFORMANCE COMPARISON

	Model	Training Accuracy (%)	Validation Accuracy (%)
1	MobileNetV2	90.342677	87.866110
2	ResNet50	95.430946	89.121342
3	EfficientNetB0	87.123573	90.794981

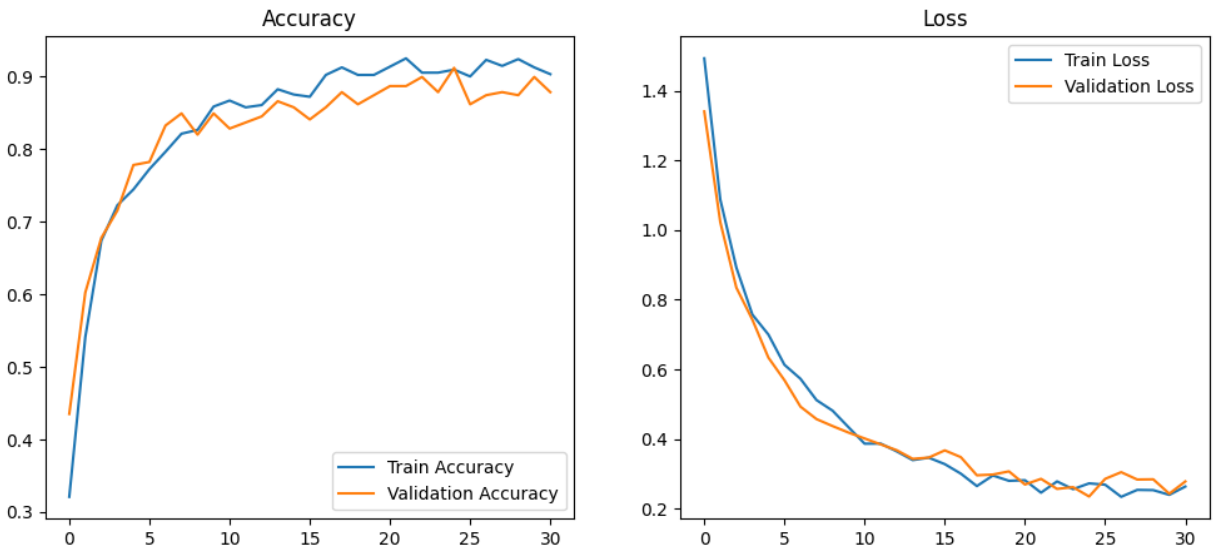

Fig3:Training and Validation Accuracy and Loss Graphs of MobileNetv2

Module 4: Face Detection and Prediction Pipeline

Objective

• To detect faces from images and apply the trained skin classification model to predict skin conditions and age.

Tasks Performed

• Implemented face detection using OpenCV and Haar Cascade classifier.

• Detected faces were cropped and resized for model input.

• Applied the trained EfficientNetB0 model to cropped face regions.

Displayed:

• Predicted skin condition

• Confidence percentage

• Predicted age

• Drew bounding boxes around detected faces and overlaid prediction results.

Deliverables

• Bounding boxes around faces

• Skin condition labels with confidence percentages

• Age prediction results

Evaluation

• Face detection accuracy was satisfactory with correct localization.

• Skin condition predictions were displayed clearly with confidence scores.

• Age prediction was successfully integrated with the detection pipeline

Fig4:Face Detection and Age Prediction

Libraries :

1. TensorFlow / Keras

Modules Used: load_img, img_to_array, ImageDataGenerator

Purpose: Loading input images into a workable format. Converting images to numerical arrays for processing. Applying augmentation techniques (rotation, zoom, shift, brightness, flip, shear), Managing preprocessing pipelines for deep-learning workflows

2. NumPy

Purpose: Numerical operations on image arrays, Array reshaping, normalization, and manipulation, Supporting backend computations for augmentation

3. Matplotlib (matplotlib.pyplot)

Purpose: Visualizing original images, Displaying augmented images in grid format. Validating dataset structure and augmentation results

4. OS Library (import os)

Purpose: Folder traversal , Dataset loading import os

5.OPEN CV

Purpose: Face detection using pre-trained Haar Cascade