Project Title:DermalScan: AI_Facial Skin Ageing Detection App

Infosys SpringBoard Virtual Internship Program

Submitted By:

Gauri Narlawar to Mr Praveen sir

Project Statement:

The objective is to develop a deep learning-based system that can detect and classify facial aging signs—such as wrinkles, dark spots, puffy eyes, and clear skin—using a pretrained EfficientNetB0 model. The pipeline includes face detection using Haar Cascades, custom preprocessing and data augmentation, and classification with percentage predictions. A web-based frontend will enable users to upload images and visualise ageing signs with annotated bounding boxes and labels.

Outcomes:

- Detect and localise facial features indicating ageing.
- Classify detected features into categories like wrinkles, dark spots, puffy eyes, and clear skin using a trained CNN model.
- Train and evaluate an EfficientNetB0 model for robust classification.
- Build a web-based frontend for uploading facial images and viewing annotated outputs.
- Interate a backend pipeline that processes input images and returns annotated results.
- Export annotated outputs and logs for documentation or analysis.

1. Introduction

Artificial intelligence and deep learning have advanced significantly in the field of computer vision. One emerging application is the automatic detection of facial aging characteristics. Aging indicators such as wrinkles, dark spots, and puffy eyes can provide insights into skin health, lifestyle habits, and dermatological conditions.
This project aims to build a deep learning–based system that automatically identifies facial aging signs from uploaded images. The final system will detect a face, classify visible signs, and present the result through a user-friendly web interface.
Milestone 1 focuses on preparing and preprocessing the dataset, which is a critical step in building a robust and accurate deep learning model.

Objective of Milestone 1

The key objectives of Milestone 1 are:
To collect, organise, and validate the dataset used for training the model.
To ensure images are properly labelled under predefined categories.
To preprocess the images for model readiness, including resizing, normalisation, and augmentation.
To encode labels into a machine-understandable format.
This ensures a strong foundation before moving to model training.

- ***Libraries Used***

NumPy:

Purpose: Numerical computing and array manipulation
Why used: Stores images as arrays (pixel matrices), performs normalisation, reshaping, and efficient mathematical operations.
Example: converting a list of images into a Numpy array for model training.

- **Panda:**

Purpose**:** Data tables and label handling
Why used**:** Helps organise labels and dataset metadata if needed for analysis.

- **OpenCV (cv2):**

Purpose: Computer vision and image processing
Role in project:
Loading images from disk
Resizing images to 224×224
Converting colour channels
Later stages: Face detection using Haar Cascade

- **Matplotlib & Seaborn**

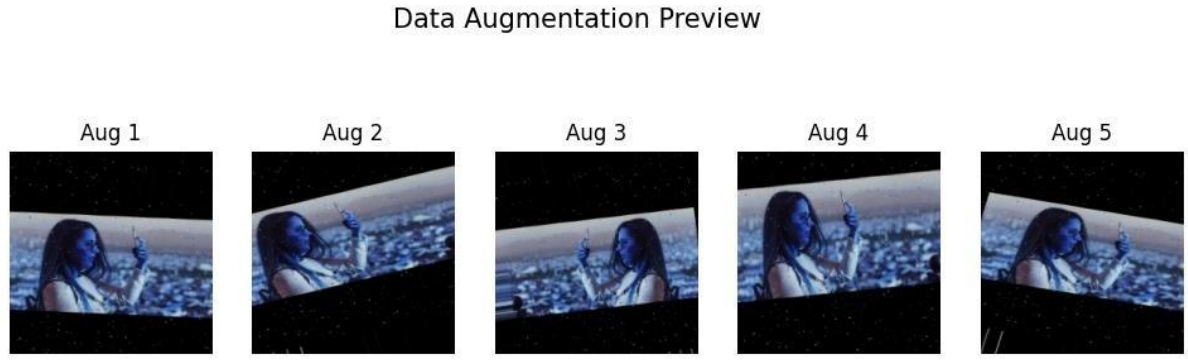
Purpose: Data visualisation and plotting
Used for:
Class distribution bar chart
Image visualization
Understanding dataset balance

- **Scikit-Learn (sklearn)**
  Purpose: Machine learning utilities
  Used for:
  Train-test split
  Label encoding
  Evaluation metrics

- **TensorFlow / Keras**
  Purpose: Deep learning framework
  Role in project:
  Image augmentation
  One-hot encoding of labels
  Later: EfficientNetB0 model training

Code:

```
import numpy as np
import random
import matplotlib.pyplot as plt
random_index = random.randrange(len(processed_images))
selected_image = processed_images[random_index]
image_array = np.expand_dims(selected_image, axis=0)
augmented_batch = [next(datagen.flow(image_array))[0] for _ in range(5)]
plt.figure(figsize=(12, 4))
for i, img in enumerate(augmented_batch):
    plt.subplot(1, 5, i+1)
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"Aug {i+1}")
plt.suptitle("Data Augmentation Preview", fontsize=15)
plt.show()
```



Conclusion of Milestone 1:

Milestone 1 successfully completes the dataset preparation and preprocessing stage. The dataset is now organised, cleaned, normalised, encoded, and augmented. This ensures that the upcoming model training phase (Milestone 2) can proceed efficiently with high-quality input data.

Modules 3

Introduction

Module 3 focuses on training a deep learning model to classify facial skin aging signs using a pretrained convolutional neural network. Transfer learning is employed to utilize previously learned visual features, enabling faster training and improved accuracy even with a limited dataset.

EfficientNetB0 Architecture

EfficientNetB0 is a state-of-the-art convolutional neural network that uses compound scaling to balance network depth, width, and resolution. It is pretrained on the ImageNet dataset and can extract rich visual features such as edges, textures, and facial patterns.

The pretrained layers serve as a feature extractor, while custom classification layers are added on top to adapt the model for facial skin aging classification.

Transfer Learning Approach

Transfer learning allows the model to reuse knowledge from large-scale datasets. In this project:

- The EfficientNetB0 base model is frozen to preserve learned features.
- Custom dense layers are added for classification into four skin aging categories.
- This approach reduces overfitting and training time.

Training Configuration

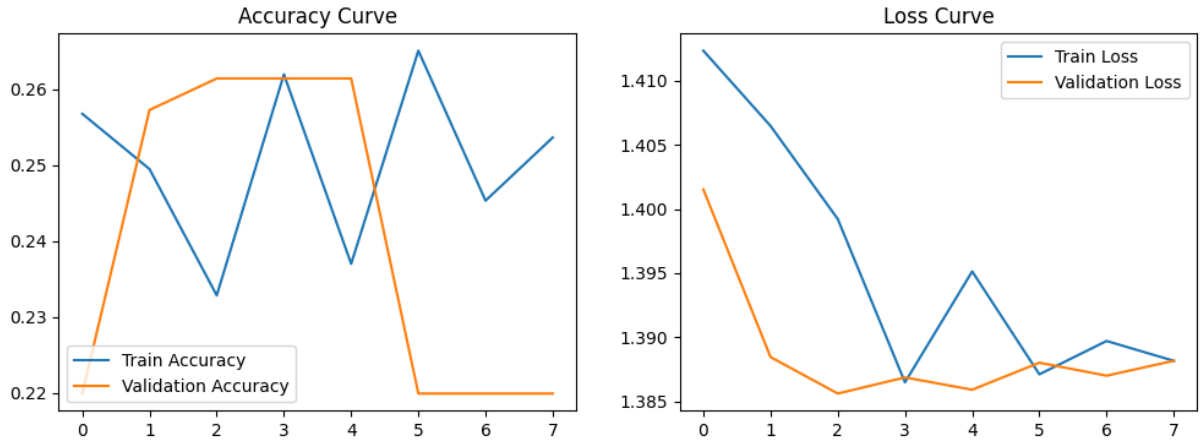
The model is trained using:

- Loss Function: Categorical Cross-Entropy, suitable for multi-class classification.
- Optimizer: Adam optimizer, which adapts learning rates for faster convergence.
- Activation Function: Softmax in the output layer to generate class probabilities.
- Validation Split: A portion of the dataset is reserved for validation to monitor performance.

Code

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Validation Accuracy")
plt.legend()
plt.title("Accuracy Curve")
plt.subplot(1,2,2)
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.legend()
plt.title("Loss Curve")
plt.show()
```

output



Model Evaluation

Model performance is evaluated using:

- Training and validation accuracy
- Training and validation loss curves
- Stability of validation accuracy over epochs

Early stopping is applied to prevent overfitting by stopping training when validation accuracy stops improving

Outcome of Module 3

The trained EfficientNetB0 model achieves high classification accuracy with stable validation performance. The final trained model is saved in .h5 format for reuse during inference and deployment.

Module 4: Face Detection and Prediction Pipeline

Introduction

Module 4 integrates computer vision techniques with the trained deep learning model to detect faces in images and predict facial skin aging signs. This module bridges the gap between model training and real-world application.

Face Detection Using Haar Cascade

Haar Cascade Classifier is a machine learning-based approach used for object detection, particularly faces. It uses Haar-like features and a cascade of classifiers to detect frontal faces efficiently.

OpenCV's Haar Cascade implementation is used due to its speed, simplicity, and real-time performance.

Prediction Pipeline

The prediction pipeline consists of the following steps:

1. Read the input image
2. Convert the image to grayscale for face detection
3. Detect face regions using Haar Cascade
4. Crop the detected face area
5. Preprocess the face (resize and normalize)
6. Predict skin aging category using the trained CNN
7. Display results on the image

Code

```
def predict_aging_clean(image_path):
img = cv2.imread(image_path)
 if img is None
print("✕ Image not found!")
return
# Convert for detection
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
 faces = face_cascade.detectMultiScale
(gray,

 scaleFactor=1.1

 minNeighbors=4,

        minSize=(80, 80)
    )# If no face detected, use whole image as fallback
    if len(faces) == 0:
        face = img.copy()
        processed_face = preprocess_face(cv2.cvtColor(face, cv2.COLOR_BGR2RGB))
        preds = model.predict(processed_face)[0]
         idx = np.argmax(preds)
        skin_label = classes[idx]
        skin_conf = preds[idx] * 10
        age_range = estimate_age(skin_label)
         text_skin = f"{skin_label} {skin_conf:.1f}%"
         text_age = f"Age: {age_range}"



        # Smaller font
        font = cv2.FONT_HERSHEY_SIMPLEX
        font_scale = 0.6
        thickness = 2
```

```
        # Put text inside top-left
        cv2.putText(img, text_skin, (10, 30), font, font_scale, (0, 255, 0), thickness)
        cv2.putText(img, text_age, (10, 55), font, font_scale, (0, 255, 0), thickness)

        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.figure(figsize=(6,6))
        plt.imshow(img_rgb)
        plt.axis("off")
        plt.show()
        return

    # For each detected face
    for (x, y, w, h) in faces:
        face_region = img[y:y+h, x:x+w]
        processed_face = preprocess_face(cv2.cvtColor(face_region, cv2.COLOR_BGR2RGB))
        preds = model.predict(processed_face)[0]

        idx = np.argmax(preds)
        skin_label = classes[idx]
        skin_conf = preds[idx] * 100
        age_range = estimate_age(skin_label)

        text_skin = f"{skin_label} {skin_conf:.1f}%"
        text_age = f"Age: {age_range}"

        # Draw green bounding box
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 3)


        # Smaller text inside the box region near top-left
        font = cv2.FONT_HERSHEY_SIMPLEX
        font_scale = 0.6
        thickness = 2


        cv2.putText(img, text_skin, (x+5, y+20), font, font_scale, (0, 255, 0), thickness)
        cv2.putText(img, text_age, (x+5, y+40), font, font_scale, (0, 255, 0), thickness)


    # Show final image
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(6,6))
    plt.imshow(img_rgb)
    plt.axis("off")
    plt.show()

dataset_root = r"DATASET-20251202T142134Z-1-001/DATASET"


cls = random.choice(os.listdir(dataset_root))
img_name = random.choice(os.listdir(os.path.join(dataset_root, cls)))
test_image = os.path.join(dataset_root, cls, img_name)


predict_aging_clean(test_image)
```

OUTPUT

Interpretation of Prediction Confidence

The confidence percentage represents the relative likelihood of the predicted class compared to other classes. It does not indicate the severity of aging but shows how confidently the model identifies a specific skin condition.

Outcome of Module 4

Module 4 successfully demonstrates an end-to-end facial skin aging detection system. It accurately detects faces, predicts aging-related skin conditions, and presents results in a visually interpretable manner