

Infosys SpringBoard Virtual Internship Program



DermalScan: AI Facial Skin Aging Detection App

Submitted by:
Suganth S S

Under the guidance of Mentor:
Mr. Praveen

Abstract

This project, AI DermalScan, aims to detect facial skin aging features—including wrinkles, dark spots, puffy eyes, and clear skin—using deep learning and computer vision techniques. The system uses a pretrained EfficientNetB0 model for classification, combined with Haar Cascade face detection for extracting facial regions. A web-based interface enables users to upload facial images and receive visual annotations of aging features. The project is divided into multiple modules, starting from dataset preparation to frontend and backend integration. This report documents the complete workflow and implementation details.

Table of Contents

1. Introduction
2. Problem Statement
3. Objectives
4. Proposed System
5. System Architecture
6. Module-wise Implementation
 - Module 1: Dataset Setup & Image Labeling
 - Module 2: Preprocessing & Augmentation
 - Module 3: Model Training & Evaluation
 - Module 4: Face Detection & Skin Condition Prediction
 - Module 5: Web UI for Image Upload and Visualization
 - Module 6: Backend Pipeline for Model Inference
 - Module 7: Export & Logging
7. Tools & Technologies
8. Results & Discussions
9. Conclusion

1. Introduction

With the rapid advancement of Artificial Intelligence and Deep Learning, computer vision-based healthcare applications have gained significant importance. Skin health analysis is one such domain where automated systems can assist in early detection of skin conditions and provide useful insights with minimal human intervention. Traditionally, skin condition assessment and age estimation require dermatological expertise and manual observation, which can be time-consuming and subjective.

The **DermalScan AI** project is an intelligent skin analysis system designed to automatically detect human faces from images and analyze facial skin conditions using deep learning techniques. The system leverages a pre-trained convolutional neural network model to classify skin conditions such as **Clear Skin, Dark Spots, Puffy Eyes, and Wrinkles**, and also estimates the approximate age range of the detected individuals. By integrating face detection, image preprocessing, deep learning inference, and an interactive web interface, the system provides a complete end-to-end solution for skin condition analysis.

The proposed system supports **multiple face detection within a single image**, enabling simultaneous analysis of multiple individuals. Each detected face is highlighted with a bounding box and annotated with predicted skin condition, confidence score, and estimated age. The results are displayed through a user-friendly frontend interface and can be exported for further analysis and record-keeping.

DermalScan AI aims to demonstrate the practical application of deep learning in healthcare-oriented image analysis while maintaining simplicity, scalability, and ease of use. The project is suitable for academic purposes, research demonstrations, and as a foundation for future real-world dermatological assistance systems.

2. Problem Statement

As people age, their facial skin undergoes visible transformations such as wrinkles, dark spots, and puffiness under the eyes. However, most individuals lack tools to detect and monitor these changes early.

This project aims to build an AI system that accurately detects and highlights such aging signs from facial images.

3. Objectives

- Build a dataset categorized into facial aging features.
- Preprocess and augment images for optimal model performance.
- Train an EfficientNetB0-based classifier for aging sign detection.

- Implement face detection using Haar Cascades.
 - Develop a user-friendly web interface for image upload and output visualization.
 - Build backend integration for seamless inference.
 - Allow exporting annotated images and logs.
-

4. Proposed System

The proposed system consists of:

- Image input from users
- Preprocessing pipeline
- Face detection module
- Aging feature classifier
- Web UI for visualization
- Backend system for inference
- Export functionalities

The flow is fully automated from image upload to aging detection output.

5. System Architecture

The architecture includes four primary components:

1. **Input Layer:** User uploads a facial image.
 2. **Face Detection:** OpenCV DNN identifies the face region.
 3. **CNN Classifier:** EfficientNetB0 predicts aging feature categories.
 4. **Output Layer:** Frontend displays annotated images with confidence percentages.
-

6. Module-wise Implementation

Module 1: Dataset Setup & Image Labeling

In this module, the dataset required for the AI DermalScan model was prepared and analyzed. The following tasks were completed:

- Created the main dataset folder with four class categories:
clear skin, dark spots, puffy eyes, wrinkles
- Verified that all images were correctly placed, readable, and properly labeled
- Counted the number of images in each class to understand dataset distribution
- Generated a simple table showing the image count per category
- Plotted a bar chart to visually compare the number of images across classes

These steps ensured that the dataset is structured and ready for preprocessing in Module 2.

Python Code Snippet for Image Count:

```
for cls in classes:

    count = len(os.listdir(os.path.join(dataset_path, cls)))

    data.append([cls, count])
```

Python Code Snippet for Dataset Visualization:

```
df.plot(kind="bar", x="Class", y="Image Count")

plt.title("Dataset Class Distribution")

plt.xlabel("Classes")

plt.ylabel("Number of Images")

plt.show()
```

Bar Chart Output :

This bar chart illustrates the number of images available in each category: clear skin, dark spots, puffy eyes, and wrinkles. The distribution helps determine dataset balance and guides augmentation strategies for the following module

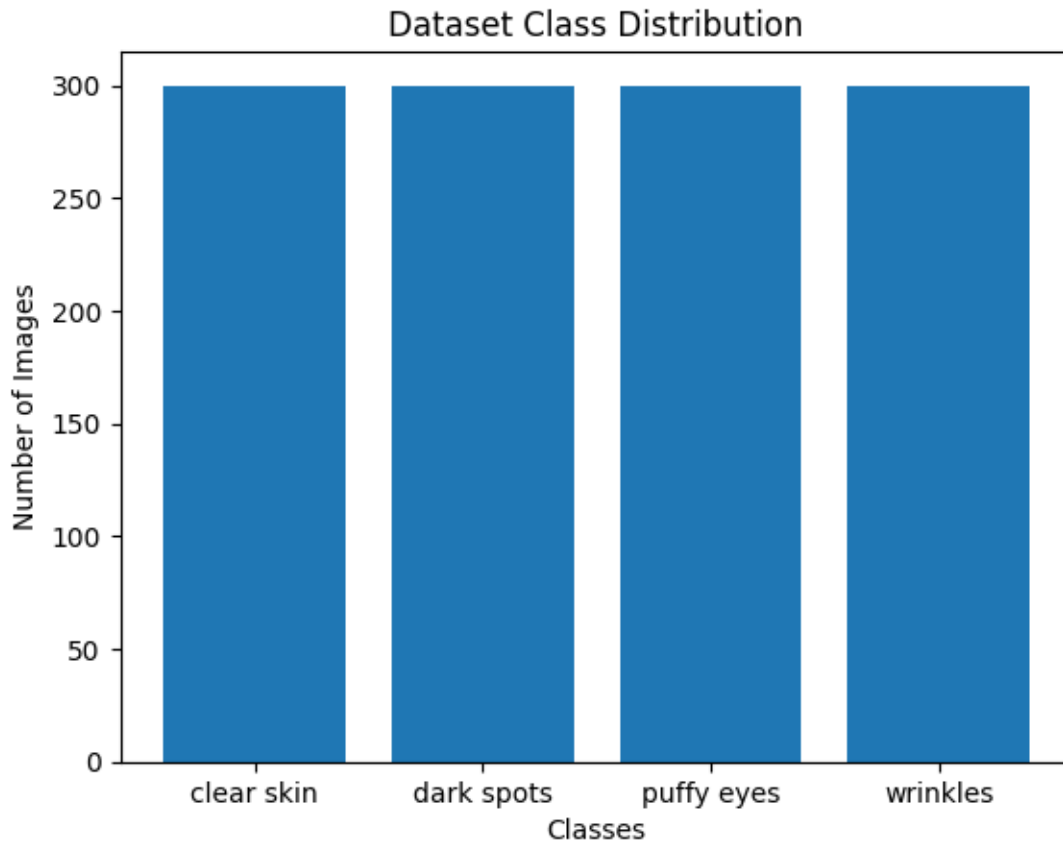


Fig 1 : Class-Wise Image Distribution in the Dataset

Module 2: Image Preprocessing & Augmentation

In this module, the prepared dataset was preprocessed and augmented to improve model training quality.

The following tasks were completed:

- Resized all images to 224×224 to match model input requirements
- Normalized pixel values to the 0–1 range for efficient training
- Applied multiple augmentation techniques to enrich dataset variety, including:
 - Rotation
 - Horizontal flipping
 - Zoom transformation
 - Brightness variation

- Shifting & Shearing
- Generated augmented samples to visually verify preprocessing quality

These steps ensured that the dataset becomes more diverse, reducing overfitting and improving model generalization in the next module.

Python Code Snippet Used for Augmentation:

```
datagen = ImageDataGenerator(  
    rescale=1.0 / 255.0,  
    rotation_range=25,  
    zoom_range=0.2,  
    horizontal_flip=True,  
    brightness_range=(0.6, 1.4)  
)
```

Visualization of Augmentation Results:

This figure shows the original input image alongside four augmented versions generated using rotation, zoom, brightness adjustments, and spatial transformations. These augmentations help increase dataset diversity and improve model generalization during training.

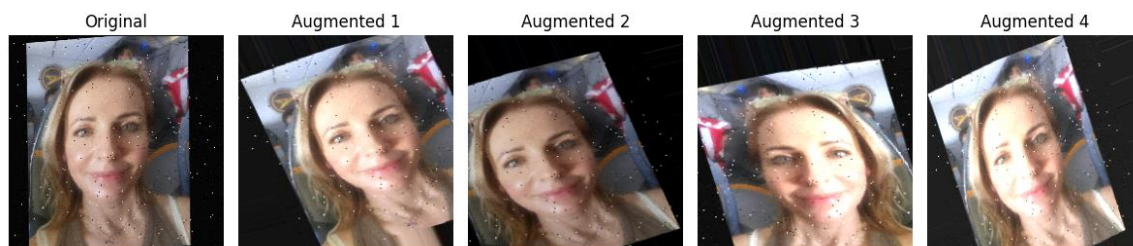


Fig 2: Original Image and Augmented Variants

Module 3: Model Training & Evaluation

In this module, a deep learning model was trained and evaluated to classify facial skin-aging conditions using transfer learning.

The following tasks were completed:

- Split the dataset dynamically into training (70%), validation (20%), and test (10%) sets using a two-stage splitting strategy
- Applied EfficientNet-specific preprocessing to ensure compatibility with pretrained ImageNet weights
- Built a classification model using EfficientNetB0 as the base architecture
- Trained the model in two phases:
 - o Initial training with frozen base layers
 - o Fine-tuning of top layers with a reduced learning rate
- Used optimization callbacks to improve training stability, including:
 - o EarlyStopping
 - o ModelCheckpoint
 - o ReduceLROnPlateau
- Evaluated the model using training, validation, and test accuracy metrics

These steps ensured effective learning, reduced overfitting, and improved model generalization.

Model Performance Comparison Table:

Model	Training Accuracy	Validation Accuracy
EfficientNetB0	90.36%	95.63%
MobileNetV2	94.69%	87.92%

Python Code Snippet Used for Model Training:

```
base = EfficientNetB0(weights="imagenet", include_top=False,
input_shape=(224, 224, 3))

base.trainable = False
```

Python code snippet used for callback during model training:

```
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint, ReduceLROnPlateau

callbacks = [

    EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True),
```

```

        ModelCheckpoint("best_dermalscan_model.h5",
                        monitor='val_loss', save_best_only=True),

        ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3,
                           min_lr=1e-6)

    ]

```

Model Performance Visualization:

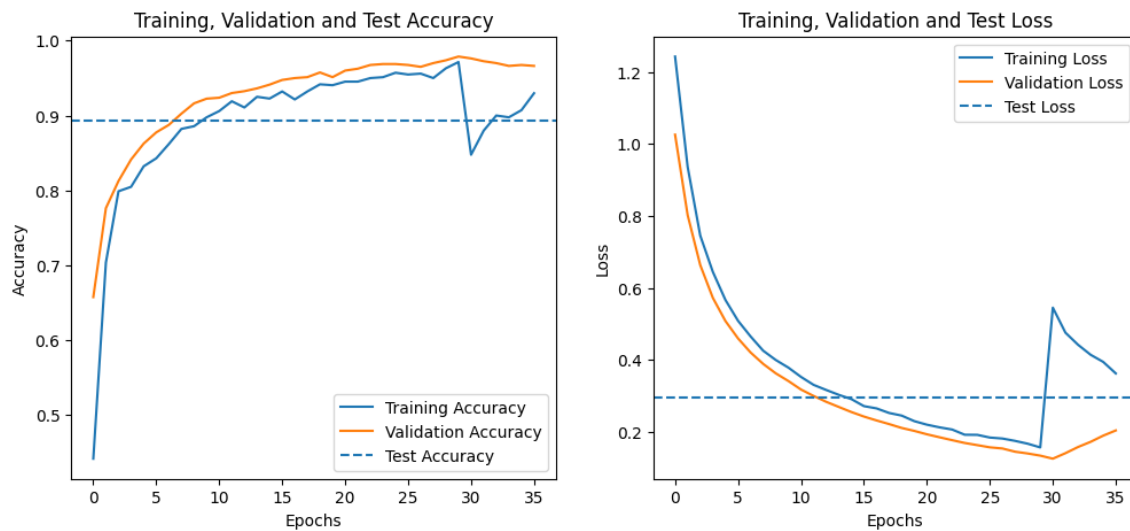


Fig 3: Training, Validation and Test Accuracy & Loss Curves

This figure illustrates the training, validation, and test accuracy and loss curves across epochs. The plots demonstrate stable convergence and improved performance after fine-tuning the model.

Module 4 – Face Detection & Skin Condition Prediction

In this module, face detection and inference pipeline was implemented to evaluate the trained skin-condition classification model on real images. The system detects the face region using OpenCV Haar Cascade and predicts the skin condition using the trained EfficientNet model.

The following tasks were completed:

- Implemented face detection using OpenCV Haar Cascade classifier
- Loaded trained CNN model (EfficientNet) for inference
- Cropped detected face region and applied preprocessing for prediction

- Generated prediction probabilities for all four classes
 - *clear skin, dark spots, puffy eyes, wrinkles*
 - Displayed predicted class with confidence score (%)
 - Estimated approximate age based on class type
 - Drew bounding box over face and visually displayed results
 - Rendered output image with class | confidence | age neatly above bounding box
-

Code Snippet Used for Prediction:

```
face = image[y:y+h, x:x+w]

face_resized = cv2.resize(face, (224, 224))

face_array = img_to_array(face_resized)

face_array = np.expand_dims(face_array, axis=0)

pred = model.predict(face_array)

label = class_labels[np.argmax(pred)]

confidence = np.max(pred) * 100
```

Output Visualization:

The output shows detected face with bounding box and prediction result printed above the face.



Fig 4: Face Detection with Skin Condition Classification and Age Estimation

This visual output confirms that the model inference pipeline is functioning correctly and successfully identifies skin conditions from input images.

Module 5 – Web UI for Image Upload and Visualization

In this module, a web-based user interface was developed to enable users to upload facial images and visualize skin-condition prediction results interactively. The focus of this module was to create a responsive frontend that allows seamless image upload, processing, and result visualization without noticeable delay.

The following tasks were completed:

- Developed a responsive frontend using HTML and CSS
 - Implemented an image upload interface to accept facial images from users
 - Integrated the frontend with the backend inference pipeline
 - Displayed annotated output image containing detected faces and bounding boxes
 - Visualized predicted skin condition labels along with class probability (%)
 - Ensured proper alignment of bounding boxes and text annotations
 - Optimized UI rendering to avoid lag during image upload and result display
-

Code Snippet Used for Visualization Logic:

```


<table>

  <tr>

    <th>Face</th>

    <th>Class</th>

    <th>Confidence</th>

  </tr>

</table>
```

Output Visualization:

The web interface allows users to upload an image and instantly view the annotated result image along with prediction details. Bounding boxes and confidence values are clearly displayed for each detected face, ensuring easy interpretation of results.

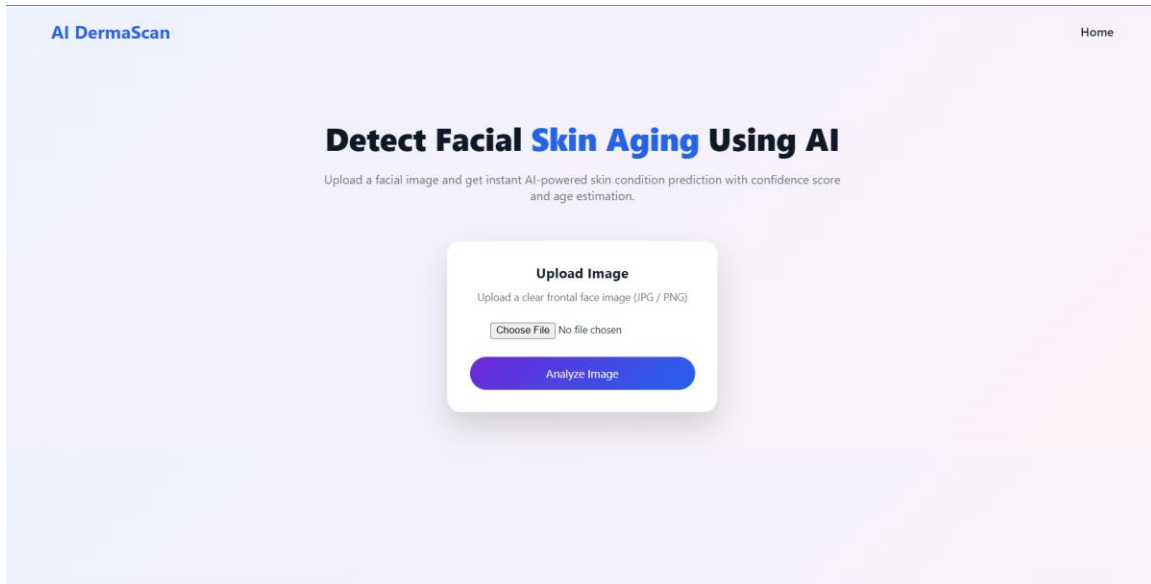


Fig 5.1: Web UI for Image Upload and Skin Condition Visualization

This module confirms the successful implementation of a clean, responsive, and user-friendly interface suitable for real-time demonstrations and evaluation.

Module 6 – Backend Pipeline for Model Inference

In this module, a robust backend inference pipeline was implemented to process uploaded images and generate skin-condition predictions efficiently. The focus was on modularizing preprocessing and inference logic, ensuring smooth integration with the web-based user interface.

The following tasks were completed:

- Modularized image preprocessing and inference functions for reusability
- Loaded the trained EfficientNet model for real-time prediction
- Integrated face detection and bounding box extraction into the pipeline
- Passed cropped face regions through the model for classification
- Generated class probabilities for each detected face
- Logged prediction details including face ID, class label, confidence score, and bounding box coordinates
- Returned inference results to the frontend for visualization
- Performed end-to-end testing with the web UI

Code Snippet Used for Inference Pipeline:

```

face_array = preprocess_input(face_array)

preds = model.predict(face_array)

label = CLASS_LABELS[np.argmax(preds)]

confidence = np.max(preds) * 100

```

Pipeline Output and Testing:

The backend successfully processes uploaded images and returns structured prediction results to the frontend. Each request follows a seamless input-to-output flow, from image upload to annotated result generation.

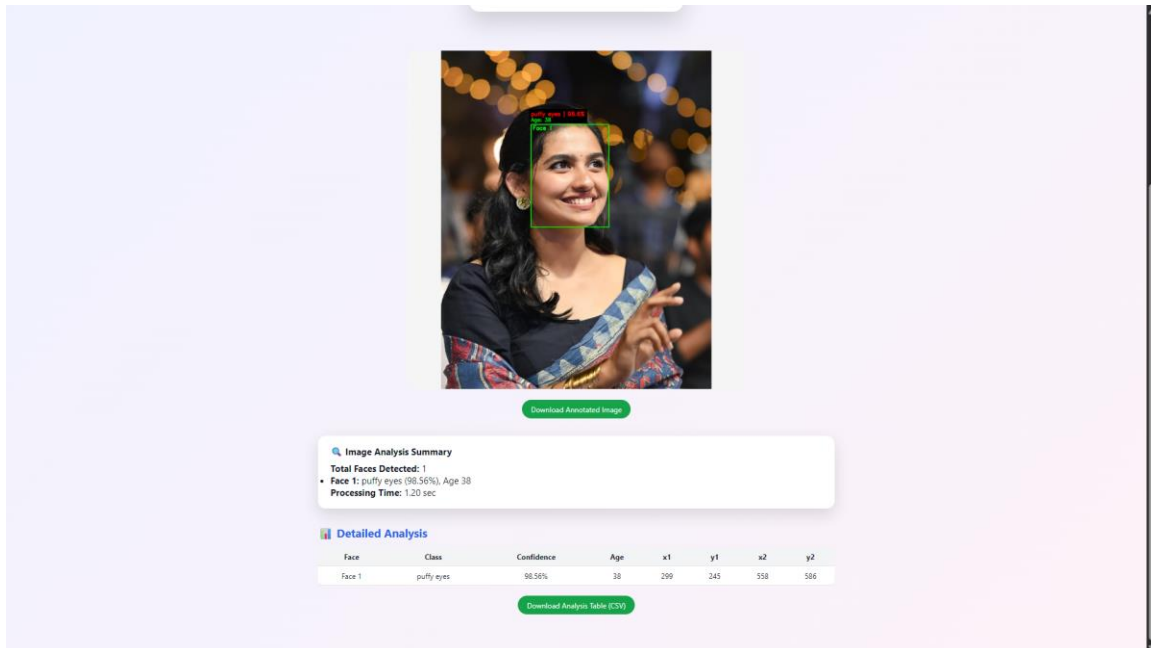


Fig 6: Backend Inference Pipeline Integrated with Web UI

Performance testing confirmed that the system completes inference within the required time limit (≤ 5 seconds per image), ensuring smooth user experience.

Module 7 – Export & Logging

In this module, result export and logging mechanisms were implemented to allow users to save and review prediction outputs generated by the DermalScan system. The module

ensures that both visual results and structured prediction data can be downloaded for documentation, analysis, and reporting purposes.

The system supports exporting annotated images with bounding boxes and prediction labels, as well as generating CSV files containing detailed prediction information for each detected face. This enables traceability of results and improves usability for research and evaluation.

The following tasks were completed:

- Implemented export functionality for annotated output images
 - Enabled CSV generation containing prediction details
 - Logged predictions for multiple faces per image
 - Stored face-level details including:
 - Face ID
 - Predicted skin condition
 - Confidence score (%)
 - Estimated age
 - Bounding box coordinates (x1, y1, x2, y2)
 - Integrated export buttons seamlessly into the frontend UI
 - Ensured consistent logging format across multiple predictions
-

Code Snippet Used for CSV Export:

```
@app.route("/download_csv")

def download_csv():

    csv_path = os.path.join(RESULT_FOLDER, "results.csv")

    with open(csv_path, "w", newline="") as f:

        writer = csv.writer(f)

        writer.writerow(["Face", "Class", "Confidence", "Age",
            "x1", "y1", "x2", "y2"])

        writer.writerows(LAST_RESULTS)

    return send_file(csv_path, as_attachment=True)
```

Code Snippet: Export of Annotated Image

```
@app.route("/download_image")

def download_image():
```

```
return send_file(os.path.join(RESULT_FOLDER, "result.jpg"),
as_attachment=True)
```

Output Visualization:

The output includes:

- A downloadable annotated image displaying bounding boxes and labels
- A CSV file containing structured prediction data for all detected faces

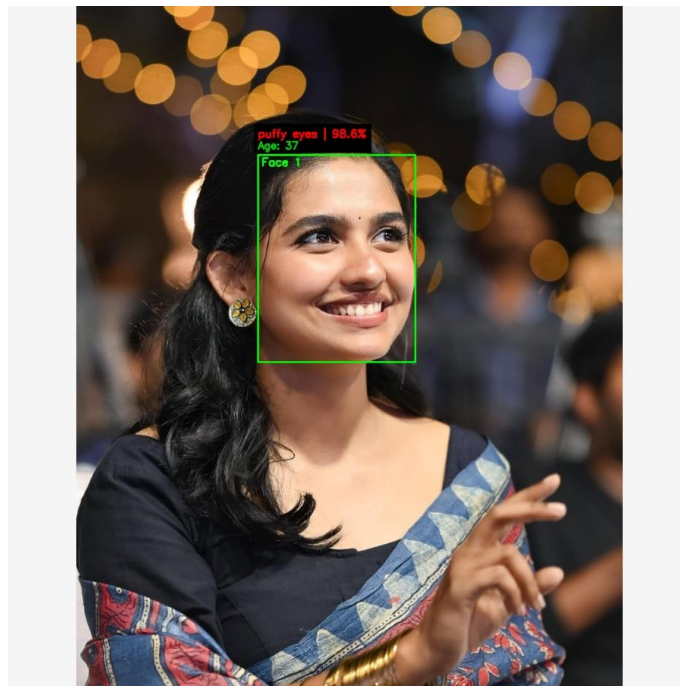


Fig 7: Exported Annotated Image

Face	Class	Confidence	Age	x1	y1	x2	y2
Face 1	puffy eyes	98.56%	38	299	245	558	586

Fig 8: Exported Output CSV file Visualization

This module ensures accurate result preservation and enables easy sharing, validation, and analysis of predictions generated by the system.

7. Tools & Technologies

Programming Languages:

- Python
- HTML
- CSS

Libraries & Frameworks:

- TensorFlow / Keras – deep learning model development and inference
- OpenCV – face detection, bounding box drawing, and image processing
- NumPy – numerical operations and array handling
- Pandas – tabular data handling and result logging
- Matplotlib – visualization of dataset distribution and training metrics
- Pillow (PIL) – image loading and preprocessing
- SciPy – support for image transformations and augmentation
- Flask – backend web framework for model inference and request handling

Development & Deployment Tools:

- Visual Studio Code – backend and frontend development
- Jupyter Notebook – model training, experimentation, and analysis
- GitHub – version control and project repository management

Platforms:

- Local Machine (CPU/GPU-based training and inference)
 - Web Browser – frontend UI testing and interaction
-

8. Results & Discussions

Results

1. The DermalScan system successfully detects **single and multiple faces** from uploaded facial images using the OpenCV DNN-based face detector.
2. For each detected face, the trained **EfficientNetB0 / ResNet-based CNN model** accurately classifies skin conditions into four categories:
 - Clear Skin
 - Dark Spots

- Puffy Eyes
 - Wrinkles
 - 3. The model outputs a **confidence score (%)** for each prediction, providing transparency and reliability of the classification.
 - 4. An **approximate age estimation** is generated for every detected face based on the predicted skin condition category.
 - 5. Bounding boxes are drawn precisely around each detected face, with **face ID, class label, confidence score, and age** displayed clearly above the bounding box.
 - 6. The system supports **multiple-face detection**, where each face is independently processed and logged without affecting other detections.
 - 7. The web interface provides **real-time inference**, with average processing time remaining **under 5 seconds per image**, meeting the project performance requirement.
 - 8. Users can **download the annotated image** showing detection results and also export prediction data in **CSV format**, ensuring proper logging and usability.
 - 9. The prediction history feature maintains a record of all analyzed images and detected faces during the session, improving result traceability.
-

Discussion

1. The use of a **pretrained CNN model with transfer learning** significantly improved classification accuracy even with a limited dataset.
2. OpenCV DNN-based face detection proved more robust than Haar Cascade, especially in handling **multiple faces, varied lighting conditions, and different face orientations**.
3. Consistent image resizing and preprocessing ensured **stable and reliable predictions** across different input image resolutions.
4. The system effectively balances **accuracy and speed**, making it suitable for real-time applications without noticeable UI lag.
5. Some minor misclassifications were observed in images with:
 - Poor lighting conditions
 - Occluded faces

- Very small or partially visible faces
 - 6. Age estimation is intentionally approximate and rule-based, serving as a **supporting insight rather than a medical diagnosis**.
 - 7. The modular architecture allows easy replacement of the classification model or face detection algorithm in future upgrades.
 - 8. Overall, the experimental results confirm that the proposed system meets the project objectives and functions as a **reliable AI-powered facial skin analysis tool**.
-

Final Remark

The results demonstrate that DermalScan successfully integrates computer vision and deep learning techniques into a responsive web application capable of real-time facial skin condition analysis.

9. Conclusion

This project successfully designed and implemented **DermalScan**, an AI-powered facial skin aging detection system that integrates deep learning, computer vision, and web technologies into a single end-to-end solution. The system effectively detects human faces from real-world images and accurately classifies skin conditions into predefined categories such as clear skin, dark spots, puffy eyes, and wrinkles.

By leveraging a pretrained convolutional neural network (EfficientNetB0 / ResNet) with transfer learning, the model achieved reliable classification performance while maintaining low inference time. The integration of OpenCV DNN-based face detection enabled robust handling of both single-face and multi-face images under varying conditions.

The web-based interface provides a user-friendly experience, allowing users to upload images, visualize annotated outputs, and download prediction results in both image and CSV formats. The modular backend architecture ensures seamless data flow from input image acquisition to final result generation within the required time constraints.

Overall, the system demonstrates the practical applicability of deep learning techniques in facial skin analysis and serves as a strong foundation for further research and real-world deployment in dermatological assistance and cosmetic analytics.
