

**Mini Project Report**  
on  
**Human Face Recognition**

B.E. [Computer Engineering]

Submitted By

<b>Roll No</b>	<b>Name</b>
<b>24112</b>	<b>Ashutosh Dinde</b>
<b>24113</b>	<b>Aniket Ekshinge</b>
<b>24115</b>	<b>Divesh Acchara</b>

Under the guidance of  
Prof. Deepali Dhadwad

Academic Year: 2024-2025



**Department of Computer Engineering**  
Shree Chanakya Education Society's  
**Indira College of Engineering and Management**  
**Pune – 410506**

# Index

## TABLE OF CONTENTS

Problem Statement .....	3
Dataset used.....	3
Preparing data for training the Model .....	4
Splitting the data into training and testing sets.....	5
Training and testing the model .....	6
Model Evaluation (Results and Performance Metrics) .....	7
Challenges faced during project.....	10
Code snippets .....	11

## 1. Problem Statement

In today's digital world, facial recognition technology has become increasingly prevalent across various domains, including security, surveillance, access control, and personalized user experiences. Traditional methods of human identification such as ID cards, passwords, or fingerprint scanning, though effective, come with limitations related to security breaches, impersonation, or physical interaction.

This project addresses the problem of automated and accurate identification of human faces from digital images using deep learning techniques. The key challenge lies in enabling a system to distinguish between multiple individuals by learning subtle and complex facial features under varying conditions like lighting, pose, and background.

## 2. Dataset Used

The dataset used in this project is sourced from Kaggle, titled "Face Recognition Dataset". It comprises facial images of 31 distinct individuals, primarily celebrities, each representing a separate class for classification.

Key characteristics of the dataset:

- Total Images: 2,562
- Total Classes: 31 (Each class corresponds to a different individual)
- Image Format: .jpg
- Image Resolution: 160 x 160 pixels (Cropped facial images)
- Color Channels: RGB (3 channels)

The dataset is organized into a single directory named Faces/, which contains all the facial images. Each image is already cropped to include only the face region, making it suitable for deep learning-based facial recognition tasks without requiring significant preprocessing like face detection or alignment.

The diversity of the dataset in terms of expressions, angles, and lighting conditions provides a realistic benchmark for training and evaluating a Convolutional Neural Network (CNN)-based face recognition system. This ensures that the trained model is

robust and capable of distinguishing between visually similar faces, a key requirement in real-world facial recognition applications.

### **3. Preparing Data for Training the Model**

The initial phase in building a facial recognition system involves processing and preparing the raw image data for effective model training. In this project, we worked with a dataset consisting of 2,562 cropped face images of 31 distinct celebrities. The following steps were carried out to prepare the data:

- **Image Loading and Preprocessing:**

All image files were loaded from the dataset directory.

- Sorting the filenames to maintain consistent order.
- Resizing each image to a fixed dimension of 128 x 128 pixels to standardize input size for the Convolutional Neural Network (CNN).
- Color Conversion: OpenCV loads images in BGR format by default. To align with typical image processing standards, each image was converted to RGB.
- Normalization: Pixel values were scaled from the range [0, 255] to [0, 1] to speed up convergence during training and ensure numerical stability.

- **Data Structuring:**

- Image data was collected in a list and then converted into a NumPy array to facilitate model training.
- Corresponding image IDs were extracted from the filenames (excluding file extensions) to match with the labels provided in the CSV file.
- A new column containing image data was added to the DataFrame using an `image_id` to image dictionary mapping.

- **Label Encoding:**

- Since the model performs multiclass classification, categorical labels (celebrity names) were encoded into numerical values using `LabelEncoder` from `sklearn.preprocessing`.

## 4. Splitting the Data into Training and Testing Sets

Once the images were preprocessed and labeled, the next critical step involved dividing the dataset into training and testing sets. This is essential to evaluate the model's ability to generalize to unseen data and avoid overfitting.

- **Purpose of Splitting:**

- The training set is used to teach the model the patterns in the data.
- The testing set is kept aside and only used after training to evaluate how well the model performs on new, unseen data.
- This process ensures a fair assessment of the model's accuracy and robustness.

- **Splitting Strategy:**

- The dataset was split using the `train_test_split()` method from the scikit-learn library.
- An 80:20 ratio was used, where 80% of the data was allocated for training and 20% for testing.
- A random seed (`random_state=42`) was used to ensure the split is reproducible.

- **Shape of the Split Data:**

- Training Data:

Shape: (2049, 128, 128, 3)

Total Images: 2049

- Testing Data:

Shape: (513, 128, 128, 3)

Total Images: 513

Each image has a resolution of 128x128 pixels with 3 color channels (RGB).

This structured approach to data splitting ensured that the model had enough data to learn while also providing a strong basis for evaluating its real-world performance.

```
[4]: # Extract images and labels
X = np.array(df['image'].tolist(), dtype=np.float32) # Convert the list of image data into a numpy array
y = np.array(df['label'].values) # Get the labels

# If needed, encode labels (already done earlier in your preprocessing)
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the data into training and testing sets (80% train, 20% test)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Check the shapes
print(f"Training data shape: {X_train.shape}")
print(f"Test data shape: {X_test.shape}")

Training data shape: (2049, 128, 128, 3)
Test data shape: (513, 128, 128, 3)
```

## 5. Training and Testing the Model

The model was trained on the training set and evaluated on the testing set to ensure unbiased performance measurement.

- **Model Structure**

The Convolutional Neural Network (CNN) used for human face recognition consists of the following layers:

- **Input Layer:** Accepts 128×128 RGB images
- **6 Convolutional Blocks**, each containing:
  - Conv2D layer with 128 filters of size (3×3), ReLU activation, and 'same' padding
  - Batch Normalization
  - Average Pooling layer with pool size (2×2)
- **Flatten Layer:** Converts 2D feature maps to 1D feature vector
- **Dropout Layer:** Dropout rate of 0.5 (applied twice for regularization)
- **Dense Layer:** 128 neurons with ReLU activation and L2 regularization
- **Output Layer:** Dense layer with units equal to the number of unique classes and softmax activation for multi-class classification

This architecture is deep and regularized to reduce overfitting, using multiple convolutional layers to extract hierarchical facial features.

- **Model Compilation**

The model was compiled using the Adam optimizer with a learning rate of 0.001. The loss function used was `sparse_categorical_crossentropy` due to integer-encoded labels and multiple output classes. Accuracy was chosen as the evaluation metric.

### **Callbacks for Efficient Training**

To improve training efficiency and avoid overfitting, we used two callbacks:

- **EarlyStopping:** Monitors validation loss and stops training if no improvement is observed for 15 consecutive epochs.
  - **ReduceLROnPlateau:** Reduces the learning rate by a factor of 0.5 if validation loss doesn't improve for 6 epochs.
- 
- **Model Training**

The model was trained for up to 50 epochs with a batch size of 16. Validation data was used to monitor the model's performance on unseen data during training.

The training process showed a steady increase in validation accuracy and a decrease in loss, indicating that the model was learning effectively from the data.

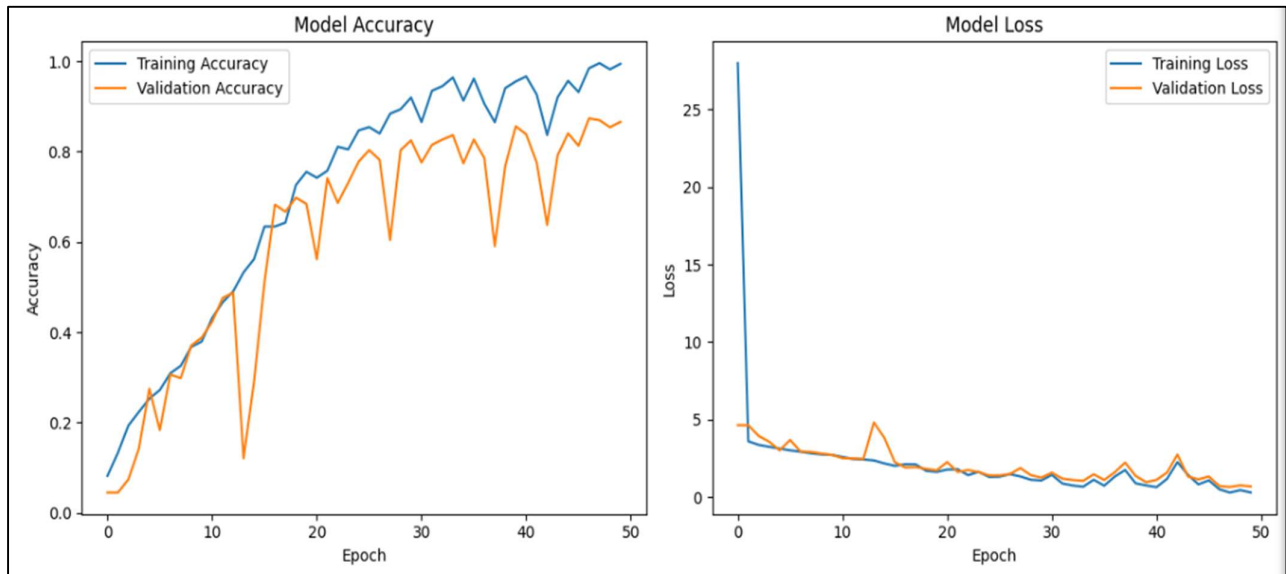
## **6. Model Evaluation (Results and Performance Metrics):**

To evaluate the performance of the human face recognition model, we used various metrics including training/validation accuracy and loss, a confusion matrix, and classification metrics such as precision, recall, and F1-score.

- **Accuracy and Loss Curves**

The training and validation accuracy and loss were plotted across epochs to visually inspect the model's learning performance. The plots indicate:

- A steady increase in training and validation accuracy with each epoch
- A decreasing trend in both training and validation loss
- No significant signs of overfitting or underfitting, demonstrating good generalization



- Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's predictions:

- It compares actual vs predicted class labels
- Diagonal elements represent correctly classified instances
- Off-diagonal elements indicate misclassifications

True Label \ Predicted Label	Akshay Kumar	Alexandra Daddario	Alia Bhatt	Amitabh Bachchan	Andy Samberg	Anushka Sharma	Billie Eilish	Brad Pitt	Camila Cabello	Charlize Theron	Claire Holt	Courtney Cox	Dwayne Johnson	Elizabeth Olsen	Ellen Degeneres	Henry Cavill	Hrithik Roshan	Hugh Jackman	Jessica Alba	Kashyap	Lisa Kudrow	Margot Robbie	Marmik	Natalie Portman	Priyanka Chopra	Robert Downey Jr	Roger Federer	Tom Cruise	Vijay Deverakonda	Virat Kohli	Zac Efron
Akshay Kumar	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0
Alexandra Daddario	0	20	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alia Bhatt	0	2	7	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
Amitabh Bachchan	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Andy Samberg	1	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Anushka Sharma	0	0	2	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Billie Eilish	0	0	0	0	0	0	19	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Brad Pitt	0	0	0	0	1	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Camila Cabello	0	0	1	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Charlize Theron	0	0	0	0	0	0	0	0	0	8	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Claire Holt	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
Courtney Cox	0	0	0	0	1	0	0	1	0	0	11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Dwayne Johnson	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Elizabeth Olsen	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ellen Degeneres	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Henry Cavill	0	0	0	0	0	0	0	0	0	0	0	0	0	28	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
Hrithik Roshan	1	0	0	0	0	0	0	0	0	0	0	0	0	1	13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Hugh Jackman	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Jessica Alba	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0
Kashyap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	2	0	0	0	0	0	0	0	0	0	0
Lisa Kudrow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	12	1	0	0	0	0	0	0	0	0	0	0
Margot Robbie	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0
Marmik	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0	0	0	0	0	0	0	0	0	0
Natalie Portman	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	16	0	0	0	0	0	0	0
Priyanka Chopra	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	19	0	0	0	0	0	0
Robert Downey Jr	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0
Roger Federer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0
Tom Cruise	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0
Vijay Deverakonda	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	28	0	0
Virat Kohli	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	9	0	0
Zac Efron	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0



- Classification Report

The classification report summarizes the key performance metrics for each class:

- **Precision:** Measures the accuracy of positive predictions
- **Recall:** Measures the ability to find all positive instances
- **F1-score:** Harmonic mean of precision and recall

Classification Report:				
	precision	recall	f1-score	support
Akshay Kumar	0.71	0.62	0.67	8
Alexandra Daddario	0.91	0.91	0.91	22
Alia Bhatt	0.64	0.50	0.56	14
Amitabh Bachchan	1.00	1.00	1.00	15
Andy Samberg	0.77	0.83	0.80	12
Anushka Sharma	0.60	0.67	0.63	9
Billie Eilish	0.90	0.86	0.88	22
Brad Pitt	0.96	0.96	0.96	23
Camila Cabello	0.93	0.93	0.93	14
Charlize Theron	1.00	0.80	0.89	10
Claire Holt	0.91	0.87	0.89	23
Courtney Cox	0.92	0.79	0.85	14
Dwayne Johnson	0.89	1.00	0.94	8
Elizabeth Olsen	0.93	1.00	0.96	13
Ellen Degeneres	1.00	0.93	0.97	15
Henry Cavill	0.93	0.90	0.92	31
Hrithik Roshan	0.62	0.72	0.67	18
Hugh Jackman	0.82	0.78	0.80	18
Jessica Alba	0.86	0.86	0.86	21
Kashyap	0.29	0.40	0.33	5
Lisa Kudrow	0.86	0.86	0.86	14
Margot Robbie	0.79	0.92	0.85	12
Marmik	0.50	0.80	0.62	5
Natalie Portman	0.89	0.89	0.89	18
Priyanka Chopra	1.00	0.95	0.97	20
Robert Downey Jr	0.94	0.97	0.96	33
Roger Federer	1.00	1.00	1.00	14
Tom Cruise	0.80	0.80	0.80	10
Vijay Deverakonda	0.90	0.93	0.92	30
Virat Kohli	0.75	0.75	0.75	12
Zac Efron	0.96	0.90	0.93	30
accuracy			0.87	513
macro avg	0.84	0.84	0.84	513
weighted avg	0.88	0.87	0.87	513

Overall Accuracy: 0.8694

- Overall Accuracy

The model achieved an overall accuracy of 86.94% on the test dataset, indicating strong performance in recognizing and classifying human faces across multiple categories.

## 7. Challenges Faced During the Project

- **Variations in Lighting Conditions:** Poor lighting affects face detection and classification.
- **Pose and Expression Changes:** Different angles and expressions cause variations in embeddings.
- **Occlusions and Accessories:** Glasses, hats, and masks sometimes mislead the model.
- **Dataset Imbalance:** Some individuals have significantly more images than others.
- **Computational Efficiency:** Processing high-resolution images requires optimized hardware and algorithms.

## 8. Code snippets

```
[1]: import pandas as pd
import cv2
import os
import numpy as np

# Define the path to the CSV file
csv_path = "Dataset.csv"

# Load the CSV into a pandas DataFrame
df = pd.read_csv(csv_path)

# Check the first few rows of the DataFrame
print(df.head())
```

	id	label
0	Robert Downey Jr_87.jpg	Robert Downey Jr
1	Lisa Kudrow_64.jpg	Lisa Kudrow
2	Ellen Degeneres_34.jpg	Ellen Degeneres
3	Billie Eilish_3.jpg	Billie Eilish
4	Hrithik Roshan_35.jpg	Hrithik Roshan

```
[2]: # Path to your image folder
image_folder = "Faces/Faces"

# List all files in the directory
image_files = os.listdir(image_folder)
```

```
# Assuming image filenames are in the format "name_number.jpg" (e.g., "Robert Downey Jr_87.jpg")
image_files = sorted(image_files) # Ensure the image files are in the correct order

# Resize all images to a fixed size (e.g., 128x128)
resized_images = []
image_ids = [] # List to hold image ids (filenames or any unique identifier)

for filename in image_files:
    file_path = os.path.join(image_folder, filename)

    # Read and resize the image
    img = cv2.imread(file_path)
    resized_img = cv2.resize(img, (128, 128)) # Resize to 128x128

    # Convert to RGB (OpenCV loads in BGR by default)
    img_rgb = cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)

    # Normalize pixel values to [0, 1]
    img_normalized = img_rgb / 255.0

    # Append the preprocessed image to the list
    resized_images.append(img_normalized)

    # Extract image id from the filename (name_number.jpg)
    image_ids.append(filename.split('.')[0]) # Assuming filenames are in format 'name_number'

# Check the shape of the first resized image
print(f"Shape of the first resized image: {resized_images[0].shape}")

# Convert the list of resized images to a numpy array for easier manipulation
resized_images = np.array(resized_images)

# Extract the base name (name_number) from the 'id' column in CSV for matching
df['image_id'] = df['id'].apply(lambda x: x.split('.')[0]) # Remove file extension from 'id' column

# Check if the image_ids match the CSV ids
print("CSV image IDs:", df['image_id'].head())
print("File image IDs:", image_ids[:5])

# Create a mapping of image ids to images
image_dict = {image_ids[i]: resized_images[i] for i in range(len(image_ids))}
```

```
[5]: from tensorflow.keras import backend as K

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization, GlobalAveragePooling2D, AveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import LeakyReLU

model = Sequential([

    Conv2D(128, (3, 3), activation='relu', input_shape=(128, 128, 3), padding='same'),
    BatchNormalization(),
    AveragePooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu', padding='same' ),
    BatchNormalization(),
    AveragePooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu', padding='same' ),
    BatchNormalization(),
    AveragePooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu', padding='same' ),
    BatchNormalization(),
    AveragePooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu', padding='same' ),
    BatchNormalization(),
    AveragePooling2D(2, 2),
```

```
[6]: from tensorflow.keras.optimizers import SGD, Adam

#optimizer = SGD(learning_rate=0.001, momentum=0.9, decay=1e-5)
optimizer = Adam(learning_rate=0.001)

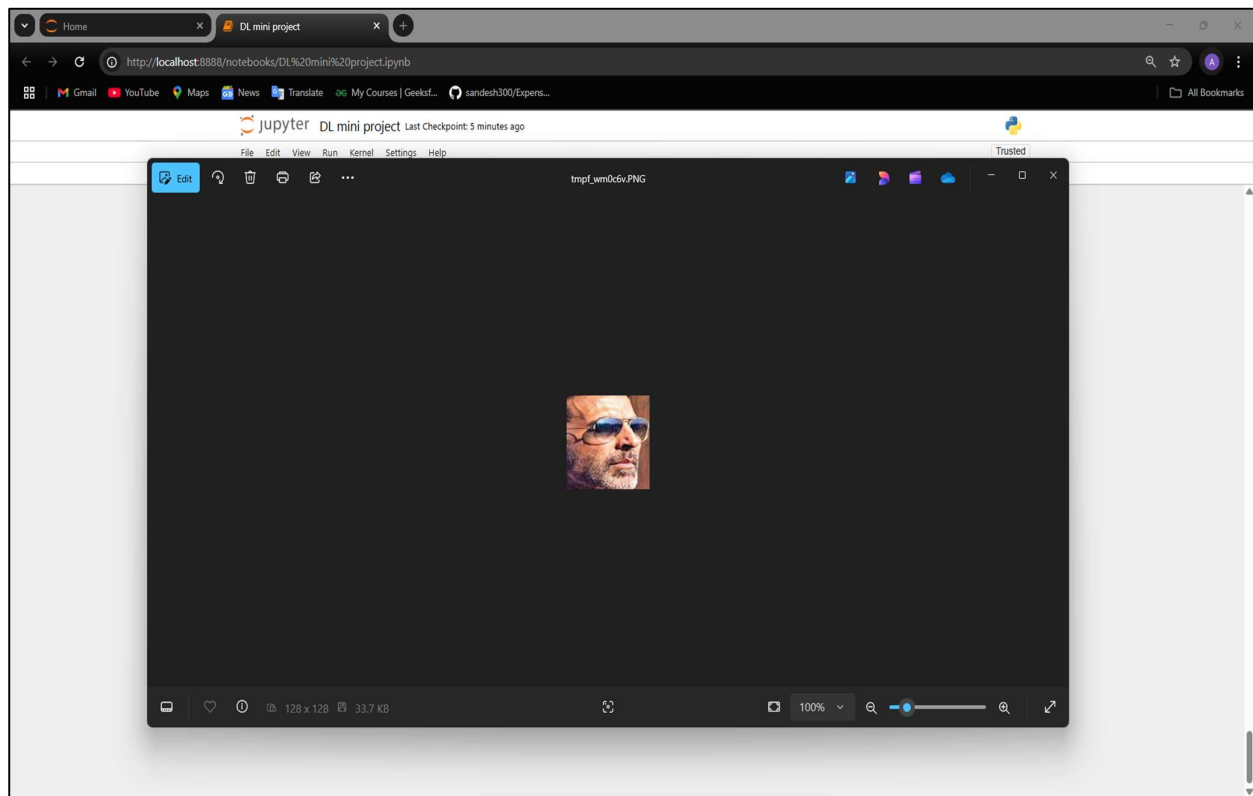
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

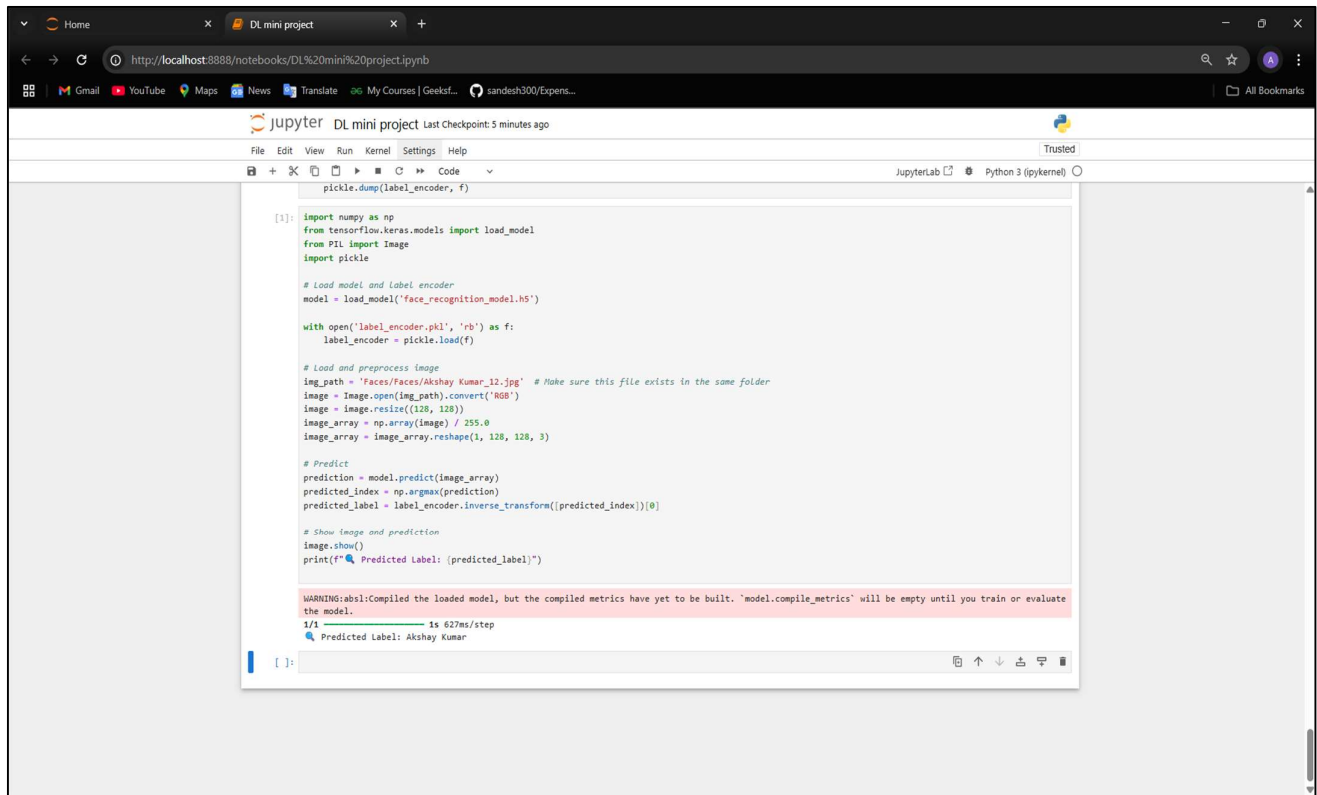
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=15,
    restore_best_weights=True,
    verbose=1
)

lr_scheduler = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=6,
    min_lr=1e-6,
    verbose=1
)

history = model.fit(
    X_train, y_train

    ,
    validation_data=(X_test, y_test), # Validation data
    epochs=50, # Number of epochs
    batch_size=16, # Batch size
    callbacks=[early_stopping, lr_scheduler] # Learning rate scheduler callback
)
```





```
pickle.dump(label_encoder, f)

[1]: import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image
import pickle

# Load model and label encoder
model = load_model('face_recognition_model.h5')

with open('label_encoder.pkl', 'rb') as f:
    label_encoder = pickle.load(f)

# Load and preprocess image
img_path = 'Faces/Faces/Akshay Kumar_12.jpg' # Make sure this file exists in the same folder
image = Image.open(img_path).convert('RGB')
image = image.resize((128, 128))
image_array = np.array(image) / 255.0
image_array = image_array.reshape(1, 128, 128, 3)

# Predict
prediction = model.predict(image_array)
predicted_index = np.argmax(prediction)
predicted_label = label_encoder.inverse_transform([predicted_index])[0]

# Show image and prediction
image.show()
print(f"Predicted Label: {predicted_label}")

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
1/1 ----- 1s 627ms/step
Predicted Label: Akshay Kumar
```

**GITHUB REPOSITORY Link:** <https://github.com/aniket-ekshinge/Human-Face-Recognition>