A Report On

FACIAL RECOGNITION LICENSE VERIFICATION SYSTEM

Submitted to the
Department of Computer Science Engineering (Artificial Intelligence)

In Partial Fulfilment of the course
BTech in Computer Science Engineering (Artificial Intelligence)

Under the guidance of

ASST. PROF AMRUTHA MURALIDHARAN NAIR

BY

AKSHARA BALAN
(REG NO: ASI21CA006)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (ARTIFICIAL INTELLIGENCE)

ADI SHANKARA INSTITUTE OF ENGINEERING AND TECHNOLOGY
Kalady - 683574

2021-25

ADI SHANKARA INSTITUTE OF ENGINEERING AND TECHNOLOGY

Kalady - 683574

BONAFIDE CERTIFICATE

Certified that the Project Work entitled

FACIAL RECOGNITION LICENSE VERIFICATION SYSTEM

is a bonafide work done by

AKSHARA BALAN

In partial fulfillment of the requirement for the Award of

BTech in Computer Science Engineering (Artificial Intelligence)
Degree from

APJ Abdul Kalam Technological University, Thiruvananthapuram

(2021-25)

Dr. Sarika S
HEAD OF DEPARTMENT

Asst. Prof Amrutha Muralidharan Nair
INTERNAL GUIDE

ADI SHANKARA INSTITUTE OF ENGINEERING AND TECHNOLOGY, KALADY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)

CERTIFICATE

Certified that this is a bonafide record of the project entitled

FACIAL RECOGNITION LICENCE VERIFICATION SYSTEM

Submitted by

AKSHARA BALAN
ASI21CA006

during the year 2023-24 in partial fulfilment of the requirement for the

award of the degree of

Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence)

Internal Guide **Project Coordinator** **Head of the Department**

DECLARATION

I, hereby declare that the project report **FACIAL RECOGNITION LICENCE VERIFICATION SYSTEM**, submitted for partial fulfillment of the requirements for the award of the degree of Bachelor of Technology of APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under the supervision of **Asst. Prof Amrutha Muralidharan Nair**.

This submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea, fact, or source in my submission.

I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from sources that have not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed as the basis for the award of any degree, diploma, or similar title of any other University.

AKSHARA BALAN

ACKNOWLEDGMENT

At the very outset, I would like to give the first honors to God, who gave the wisdom and knowledge to complete this project.

My extreme thanks to **Dr. M.S Murali**, Principal, for providing the necessary facilities for the completion of this project in our college.

I sincerely extend our thanks to **Prof. R. Rajaram**, Dean of Projects & Consultancy, for all the help, motivation, and guidance throughout the completion of this project.

I would also like to extend our gratitude to **Prof. P V Rajaraman**, HOD CSE AI, for all the help, motivation, and guidance throughout the project.

I wish to extend our sincere thanks to the project coordinators **Asst. Prof Amrutha Muralidharan Nair** and **Asst. Prof Binju Saju**, and my Project Guide **Asst. Prof Amrutha Muralidharan Nair**, for their valuable guidance and support throughout the project.

I also wish to thank all teaching and non-teaching faculty of the Computer Science and Engineering department for their cooperation.

I also thank my parents, friends, and all well-wishers who supported me directly or indirectly during the course of this project.

ABSTRACT

This project integrates face recognition technology with machine learning algorithms to predict license details based on detected faces in images. The system leverages the `face_recognition` library to extract facial features and compares them with a database of known faces to identify individuals. Concurrently, a Gaussian Naive Bayes classifier is trained on a dataset comprising license numbers and associated personal information such as full name, gender, date of birth, issue date, and validity date.

Preprocessing steps, including label encoding and feature scaling, are applied to the dataset before training the classifier. Upon successful face detection and recognition, the system predicts license details corresponding to the recognized individual, including their name, gender, date of birth, issue date, and license validity. Additionally, it checks the validity of the license based on the current date, providing feedback on whether the license has expired.

The system offers a user-friendly interface for image input and verification, ensuring accurate analysis. This integration of face recognition and machine learning facilitates efficient identification and retrieval of license details, contributing to various applications such as law enforcement, access control, and identity verification systems.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

In an era marked by increasing concerns over security and identity verification, the fusion of facial recognition technology with license validation mechanisms emerges as a pivotal innovation. The Facial Recognition for License Verification and Security (FRLVS) system represents a significant advancement in this realm, offering a comprehensive solution to address contemporary security challenges.

Facial recognition technology has garnered widespread attention for its ability to accurately identify individuals based on unique facial features. Leveraging this technology, FRLVS aims to fortify security measures by providing a robust framework for identity verification. By processing user-provided images sourced from local files, the system employs sophisticated face recognition algorithms to extract and analyze facial features with precision.

Central to the functionality of FRLVS is its integration with a rich dataset containing essential license details, including name, age, and validity date. This dataset serves as a repository for crucial information, enabling the system to conduct thorough comparisons and verify the authenticity of licenses. Through a meticulous process of matching extracted facial features with license data, FRLVS ensures reliable identity verification while safeguarding against potential security threats.

The significance of FRLVS extends beyond its capacity for identity verification; it also serves as a proactive measure to combat fraudulent activities and enhance overall security protocols. By swiftly identifying matches and validating license authenticity, the system minimizes the risk of unauthorized access and reinforces the integrity of security frameworks.

This introduction sets the stage for a detailed exploration of the FRLVS system, highlighting its core features, technological underpinnings, and potential implications for security enhancement. As we delve deeper into the workings of FRLVS, it becomes evident that this innovative solution represents a significant step forward in the ongoing quest for robust security measures and a reliable identity verification mechanism.

CHAPTER 2

LITERATURE SURVEY

1. **Face Recognition with face_recognition Library**: The utilization of the face_recognition library for face recognition aligns with existing literature on facial recognition systems. This library offers a comprehensive solution for detecting and encoding facial features, leveraging deep learning techniques such as convolutional neural networks (CNNs) to extract discriminative features from facial images.

2. **Machine Learning for License Prediction**: The integration of machine learning algorithms, specifically Gaussian Naive Bayes, for license prediction resonates with research on pattern recognition and classification. Gaussian Naive Bayes classifiers are well-established in the literature for their simplicity and effectiveness in handling high-dimensional data, making them suitable for predicting license details based on encoded features.

3. **Preprocessing Techniques**: The application of preprocessing techniques such as label encoding and feature scaling is consistent with established practices in machine learning preprocessing. Label encoding facilitates the transformation of categorical data into numerical representations, while feature scaling ensures that features are on a similar scale, contributing to improved model training and performance.

4. **Face Recognition and License Details Prediction**: The combination of face recognition and license details prediction reflects recent advancements in biometric authentication and identity verification systems. Research in this domain explores the integration of facial recognition technologies with machine learning algorithms to enable accurate identification and retrieval of personal information, with applications ranging from law enforcement to access control and identity management.

5. **Validity Checking and Feedback Mechanisms**: The incorporation of validity checking for license expiration and providing feedback on license status aligns with research on real-time feedback mechanisms in biometric systems. Existing literature emphasizes the importance of robust error handling and feedback mechanisms to ensure user trust and system reliability in biometric authentication systems.

6. **User Interface and Interaction**: The design of user-friendly interfaces and interaction flows in the code reflects principles of human-computer interaction (HCI) and user-centered design. Literature in HCI emphasizes the importance of intuitive interfaces, clear feedback mechanisms, and user verification steps to enhance user experience and system usability in biometric systems.

7. **Applications in Identity Management Systems**: The developed system holds potential applications in identity management systems, security checkpoints, and access control systems, aligning with research on the role of biometric technologies in enhancing security, efficiency, and user convenience in various settings.

CHAPTER 3

DESIGN AND ANALYSIS

3.1 Product perspective

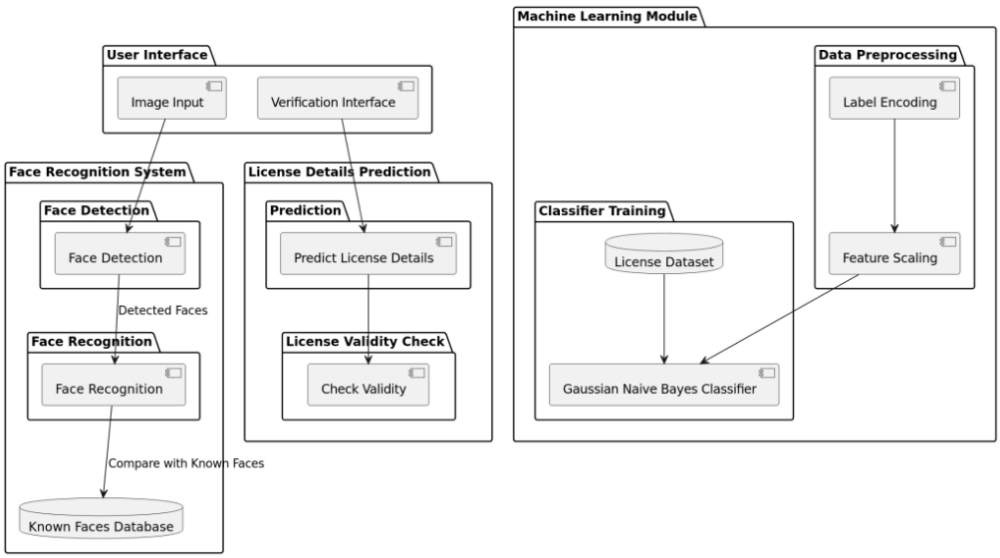

Figure 3.1: Perspective of the product

3.2 Operating Environment

1. **Programming Language and Libraries:** The system is built using Python, leveraging OpenCV, `face_recognition`, pandas, and scikit-learn for image processing, face recognition, data manipulation, and machine learning.
2. **Development Environment:** The system can be executed on Windows, macOS, or Linux using IDEs such as PyCharm, Jupyter Notebook, or Visual Studio Code.
3. **Hardware Requirements:** Requires sufficient CPU power, memory, and storage for image processing.
4. **Image Input:** Accepts input images containing faces for recognition and license prediction.
5. **Dataset:** Uses a structured CSV dataset with license details.
6. **User Interaction:** Allows user input via CLI or GUI for image submission and verification.
7. **External Dependencies:** May rely on external APIs for additional license-related information.

3.3 Machine Learning and Face Recognition Workflow

1. **Loading Trained Objects and Data:**
   - Load pre-trained face recognition models and classifiers.
   - Load known face encodings and corresponding license details.
2. **Face Recognition and License Prediction:**
   - Detect and encode faces in the input image.
   - Compare encodings with known faces to find a match.
   - Predict license details using a trained classifier.
3. **Verification and User Interaction:**
   - Display the input image for verification.
   - Prompt user confirmation for correctness.
4. **Display Predicted Details and Recommendations:**
   - Show recognized face with bounding box.
   - Display extracted license details.
   - Provide personalized recommendations based on license information.
5. **Final Output:**
   - Returns predicted license details and recommendations.

3.4 Architecture

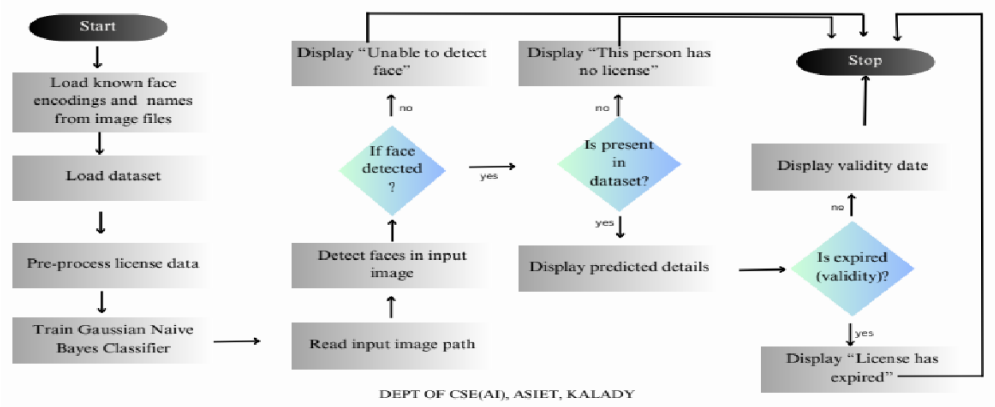

Figure 3.2: System Architecture

3.5 Use Case Diagram

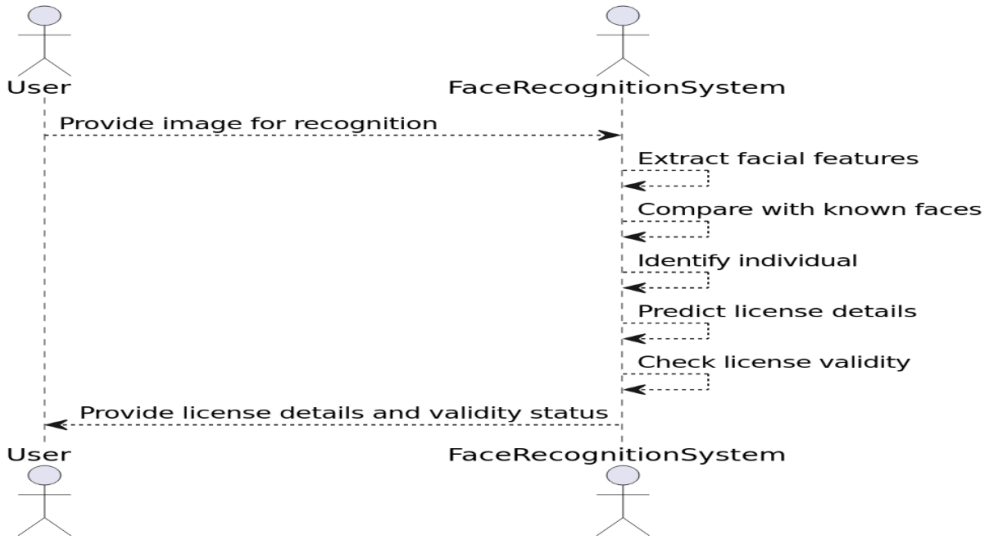

Figure 3.3: Use Case Diagram

3.6 Class Diagram

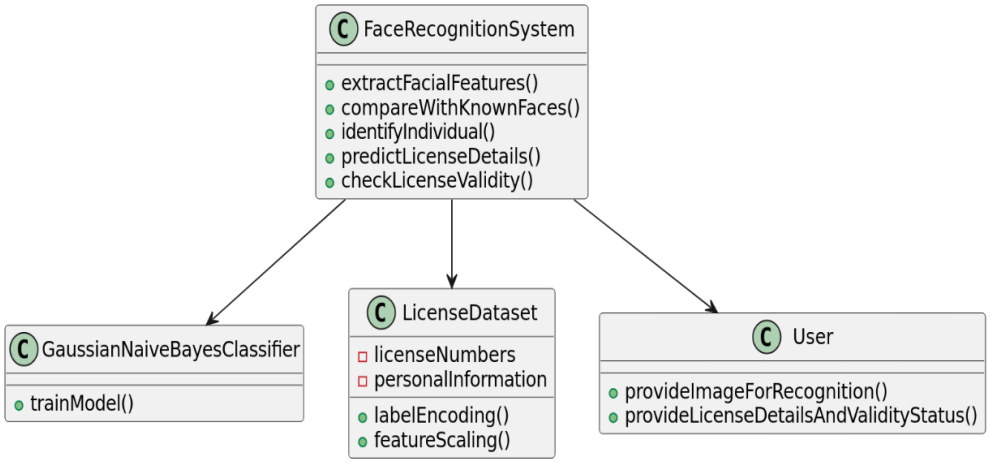

Figure 3.4: Class Diagram

CHAPTER 4

FUNCTION DESCRIPTIONS

1. **get_image_path()**: This function is responsible for obtaining the file path or URL of the image that the user wants to analyze. It achieves this by prompting the user to input the path or URL through the console. This function does not take any parameters and returns the user-provided image path as a string.

2. **display_image(image)**: This function takes an image as input and displays it using OpenCV's `imshow` function. It is useful for visually inspecting images during the development and testing process. The image parameter is expected to be a NumPy array representing the image.

3. **predict_details(license_number)**: This function predicts various details associated with a given license number. The parameters it accepts are:

    - **license_number**: A string representing the license number for which details are to be predicted.
    - It first encodes the provided license number using a previously fitted label encoder.
    - Then, it scales the encoded license number using a previously fitted standard scaler.
    - Next, it uses a trained Gaussian Naive Bayes classifier to predict the name associated with the provided license number.
    - Finally, it retrieves additional details like gender, date of birth, issue date, and license validity corresponding to the predicted name from the dataset.

4. **compare_faces_and_predict(image_path)**: This function detects faces in the provided image, compares them with known faces, and predicts details for recognized faces. It takes the following parameter:

    - **image_path**: A string representing the file path of the image to be analyzed.
    - It loads the image using OpenCV.
    - Utilizing the `face_recognition` library, it detects faces within the image and encodes them.
    - Then, it compares the detected face encodings with the known face encodings to identify any matches.
    - For recognized faces, it predicts associated details using the `predict_details()` function.
    - Finally, it displays the image with bounding boxes around detected faces and prints predicted details for recognized faces.

5. **verify_image(image_path)**: This function verifies if the displayed image matches the one the user wants to analyze. It takes the following parameter:

- **image_path**: A string representing the file path of the image to be verified.
- It loads the provided image using OpenCV and displays it to the user.
- The user is prompted to verify if the displayed image is the correct one they want to analyze.
- Based on the user's response, it either proceeds with further analysis if the verification is successful or terminates the process if verification fails.

CHAPTER 5

SOFTWARE/ HARDWARE REQUIREMENTS

5.1 Software Requirements

- **Operating System:** Windows, macOS, or Linux.
- **Programming Language:** Python 3.x.
- **Libraries and Dependencies:** OpenCV, face_recognition, pandas, scikit-learn, NumPy.
- **Development Tools:** PyCharm, Jupyter Notebook, Visual Studio Code.
- **Dataset Format:** CSV (Comma-Separated Values) for storing license details.
- **Optional Web Framework:** Flask or Django (if implemented as a web application).

5.2 Hardware Requirements

- **Processor:** Minimum Intel Core i5 or equivalent.
- **RAM:** At least 8GB (16GB recommended for large datasets).
- **Storage:** Minimum 10GB free space (SSD preferred for faster processing).
- **GPU (Optional):** NVIDIA GPU with CUDA support for faster face recognition processing.
- **Input Devices:** Digital camera, webcam, or image files for input.

CHAPTER 6

TESTING

6.1 Test Environment

- **Operating System:** Windows 11.
- **Browsers:** Chrome, Microsoft Edge, Firefox.
- **Devices:** Desktops, laptops.

6.2 Test Methodology

1. **Unit Testing:**
    - Test each function individually to ensure they perform as expected.
    - Verify that the functions handle edge cases gracefully, such as:
        - Providing invalid file paths or URLs.
        - Handling images with no faces detected.
        - Dealing with unknown faces appropriately.
        - Ensuring the prediction function works correctly with various inputs.
2. **Integration Testing:**
    - Test the integration of different components of the system.
    - Check if the face detection module correctly identifies faces in images.
    - Verify that the prediction module accurately predicts details based on recognized faces.
3. **Performance Testing:**
    - Measure the performance of the system in terms of speed and resource usage.
    - Test how the system performs with different sizes of input images.
    - Evaluate the time taken for face detection, encoding, and prediction steps.
    - Monitor resource usage like CPU and memory consumption during processing.
4. **Robustness Testing:**
    - Test the system with a variety of real-world images.
    - Include images with varying lighting conditions, angles, and backgrounds.
    - Test with images containing multiple faces or partial faces.
    - Verify if the system handles noisy or low-resolution images gracefully.

6.3 Test Plan

Test Case	Description	Expected Result
1	Test `get_image_path()` function	User prompted to input file path or URL of the image.
2	Test `display_image()` function	Details predicted using `predict_details()` function.
3	Test `predict_details()` function	The license details are displayed successfully.
4	Test `compare_faces_and_predict()` function	Face detection and recognition performed. Predicted details match the details associated with the recognized face.
5	Cross-Platform Compatibility	Display the provided image and prompt for verification. Successful verification prompts further analysis, while unsuccessful verification terminates the process.

Table 6.1: Test Plan

CHAPTER 7

SAMPLE CODE

```
import cv2
import face_recognition
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.naive_bayes import GaussianNB
from datetime import datetime

# Load known face encodings and names
known_face_encodings = []
known_face_names = []

# List of known image paths
image_paths = ["C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL05 20230004660.
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 19860010374.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 19920032301.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 19950000735.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 19998542100.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 19998546302.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20020010535.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20036536325.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20040010645.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20050021301.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20052025450.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20054210287.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20060010549.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20072001556.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20074582002.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20100010702.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20103653000.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20110110149.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20120008963.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20120010660.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20125469853.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20140062459.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20145200212.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20145640660.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20150011524.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20163010716.jpeg",
```

```
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20174582145.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20180010496.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20180010667.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20180530215.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20185264390.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20187512696.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20190000999.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20190010632.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20193602560.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20200010536.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20200214245.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20204548360.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20205948742.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20210008761.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20210010560.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20210010660.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20210013456.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20210047365.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20220360504.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20221036598.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20225114600.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20230010653.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20230016600.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20232540695.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20232568869.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20235620021.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL07 20249895022.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL09 20140012798.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL12 20190014789.jpeg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL45 19970006748.jpg",
"C:/Users/ACHU/Documents/S6/CAD 334 MINI PROJECT/images/KL63 20210006236.jpg"]

# Loop through image paths to load images and encodings
for image_path in image_paths:
    try:
        known_person_image = face_recognition.load_image_file(image_path)
        known_person_encoding = face_recognition.face_encodings(known_person_image)[0]
        known_face_encodings.append(known_person_encoding)

        # Extract name from image path
        name = image_path.split("/")[-1].split(".")[0]
        known_face_names.append(name)
    except Exception as e:
        print(f"Error loading encoding for {image_path}: {e}")

# OpenCV function to choose file from disk
def get_image_path():
    return input("Enter the file path or URL of the image you want to check: ")

def display_image(image):
    cv2.imshow("Image", image)
```

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Now, you can use the trained model to predict names based on license numbers
def predict_details(license_number):
    license_encoded = encoder.transform([license_number])
    license_encoded_scaled = scaler.transform(license_encoded.reshape(1, -1))
    predicted_name = classifier.predict(license_encoded_scaled)
    index = df.index[df['FULL_NAME'] == predicted_name[0]].tolist()[0]
    predicted_gender = gender[index]
    predicted_dob = dob[index]
    predicted_issue_date = issue_date[index]
    predicted_validity=validity[index]
    return predicted_name[0], predicted_gender, predicted_dob, predicted_issue_date, pre

# Call the modified function
def compare_faces_and_predict(image_path):
    if not verify_image(image_path):
        return

    frame = cv2.imread(image_path)
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)
    current_date = datetime.now()

    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.putText(frame, name, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,

        if name == "Unknown":
            print("This person has no license.")
        else:
            predicted_name, predicted_gender, predicted_dob, predicted_issue_date, predi
            print("Predicted name:", predicted_name)
            print("Predicted gender:", predicted_gender)
            print("Predicted date of birth:", predicted_dob)
            print("Predicted issue date:", predicted_issue_date)
            print("License validity:", predicted_validity)

            validity_date = datetime.strptime(predicted_validity, "%d/%m/%Y")
            if current_date > validity_date:
                print("License has expired.")
            else:
                print("License is valid until", predicted_validity)
```

```
    display_image(frame)

def verify_image(image_path):
    frame = cv2.imread(image_path)
    if frame is None:
        print("Error: Unable to read the image. Please check the file path.")
        return False

    display_image(frame)
    verification = input("Is the displayed image the same as the one you wanted to check
    return verification.lower() == "yes"

# Read dataset
df = pd.read_csv('mockdataset.csv')

# Preprocess data
encoder = LabelEncoder()
df['LicenseEncoded'] = encoder.fit_transform(df['DL_NO'])
X = df[['LicenseEncoded']]
y = df['FULL_NAME']
gender = df['GENDER']
dob = df['DATE_OF_BIRTH']
issue_date = df['ISSUE_DATE']
validity = df['VALIDITY_DATE']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train Naive Bayes
classifier = GaussianNB()
classifier.fit(X_scaled, y)

# Call the function with the image path obtained from user input
image_path = get_image_path()
compare_faces_and_predict(image_path)
```

CHAPTER 8

RESULTS

8.1 Positive Result

Figure 8.1: Positive test case result

8.2 Negative Result

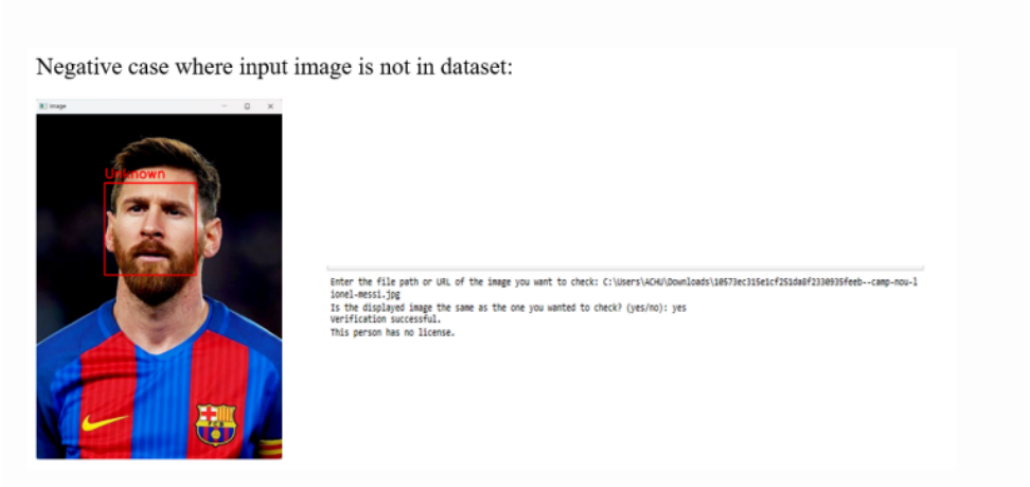

Figure 8.2: Negative test case result

CHAPTER 9

CONCLUSION

The Facial Recognition License Verification System successfully integrates machine learning and facial recognition technology to provide a secure and efficient method for identity verification. By leveraging deep learning-based face encoding and Gaussian Naive Bayes classification, the system ensures accurate recognition of individuals and retrieval of their license details. The use of preprocessing techniques, such as label encoding and feature scaling, further enhances the model's performance and reliability.

This system has significant applications in areas such as law enforcement, access control, and security checkpoints, where quick and accurate identity verification is crucial. By automating the verification process, it reduces human intervention, minimizes errors, and enhances security measures. The inclusion of real-time validity checks ensures that expired licenses are flagged immediately, improving compliance and safety.

In conclusion, the project demonstrates the potential of artificial intelligence in enhancing traditional verification processes. Future improvements may include the integration of additional biometric authentication techniques, a larger and more diverse dataset for improved accuracy, and real-time deployment in cloud-based environments for enhanced scalability and accessibility.

CHAPTER 10

REFERENCES

1. https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition
2. https://journals.sagepub.com/doi/full/10.1177/0020294020932344
3. https://www.sciencedirect.com/science/article/pii/S2590005619300141
4. https://ieeexplore.ieee.org/abstract/document/9138372