

Acknowledgment

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our mentor, **Dr. Sharmishta Desai**, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of the staff of MIT WPU, who gave the permission to use all required equipment and the necessary materials to complete the task. A special thanks goes to my team mates, who helped me enormously to assemble the parts and gave suggestion about the task of using the techniques of measurements.

I have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

I would also like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I wish to thank my friends for their support and encouragement throughout my study.

Name of Students

Krishnaraj Thadesar, 1032210888
Parth Zarekar, 1032210846
Sourab Karad, 1032211150
Saubhagya Singh, 1032211144

Abstract

The Attendance-Assistant project introduces an innovative, cross-platform mobile application designed to automate attendance tracking in educational institutions using advanced facial recognition technology. The system leverages Flutter for the frontend, ensuring a seamless and intuitive user experience, while the backend is powered by FastAPI and MongoDB, providing robust, secure, and scalable data management. This comprehensive solution streamlines the entire attendance workflow, encompassing user authentication, face-encoding management, real-time attendance marking, and detailed reporting.

This report documents the systematic approach undertaken to evaluate and compare various face recognition algorithms, focusing on their precision metrics, computational efficiency, and real-world applicability. After rigorous testing, the ResNet-based implementation in Python, utilizing the *face_recognition* library, was selected for its superior accuracy and reliability. The report further elaborates on the system architecture, database schema, and core functionalities, highlighting how the Attendance-Assistant enhances operational efficiency, accuracy, and user convenience compared to traditional attendance methods.

The project also explores the integration of IoT devices, such as Raspberry Pi, to enable real-time facial recognition in classrooms and other institutional settings. By combining cutting-edge technologies with a user-centric design, the Attendance-Assistant aims to revolutionize attendance management, offering a scalable and efficient solution for modern educational institutions.

Keywords

Facial Recognition, Attendance Automation, Flutter, FastAPI, MongoDB, ResNet, Educational Institutions, IoT, Raspberry Pi, Backend Development, Frontend Development, Data Integrity, Firebase, Precision Metrics, Attendance Management System

List of Figures

7.1	Cropped Faces using OpenCV Haar Cascades	20
7.2	Cropped Faces using OpenCV Haar Cascades	20
7.3	Cropped Faces using OpenCV Haar Cascades	21
7.4	Krishnaraj's Segregated Images	21
7.5	Saubhagya's Segregated Images	22
7.6	Parth's Segregated Images	22
7.7	Sourab's Segregated Images	23
7.8	Abhijeet's Segregated Images	23
7.9	Results identifying 3 of the 4 faces. Empirical results show that the model is working, with accuracy of around 75 %	24
7.10	Accuracy per Person (Class) - LBPH Face Recognizer	30
10.1	Confusion Matrix of LBPH Face Recognizer	34
10.2	Accuracy per Person (Class) - LBPH Face Recognizer	34
10.3	Accuracy per Person (Class) - LBPH Face Recognizer	35
10.4	Confusion Matrix of LBPH Face Recognizer	36
10.5	Accuracy per Person (Class) - LBPH Face Recognizer	36
10.6	Accuracy per Person (Class) - LBPH Face Recognizer	37
10.7	Confusion Matrix of LBPH Face Recognizer	38
10.8	Accuracy per Person (Class) - LBPH Face Recognizer	38
10.9	Accuracy per Person (Class) - LBPH Face Recognizer	39

List of Tables

7.1 Training Data Images	18
------------------------------------	----

Contents

1	Introduction	8
1.0.1	Problem Statement	8
1.0.2	Need of the Project	8
2	Literature Survey	9
2.1	Paper 1	9
2.1.1	Positives and Learnings from this Paper	9
2.1.2	Identified Research Gaps	9
2.2	Paper 2	9
2.2.1	Positives and Learnings from this Paper	9
2.2.2	Identified Research Gaps	10
2.3	Paper 3	10
2.3.1	Positives and Learnings from this Paper	10
2.3.2	Identified Research Gaps	10
2.4	Paper 4	10
2.4.1	Positives and Learnings from this Paper	10
2.4.2	Identified Research Gaps	11
2.5	Paper 5	11
2.5.1	Positives and Learnings from this Paper	11
2.5.2	Identified Research Gaps	11
3	Problem Statement	12
3.1	Project Scope	12
3.2	Project Assumptions	12
3.3	Project Limitations	12
3.4	Project Objectives	12
4	Project Requirements	13
4.1	Resources	13
4.1.1	Human Resources	13
4.1.2	Reusable Software Components	13
4.2	Requirements	13
4.2.1	Software and Hardware Requirements	13
4.3	Risk Management	13
4.4	Functional Specifications	13
4.4.1	Interfaces	13
4.4.2	Interactions	13
5	System Analysis Proposed Architecture	14
5.1	Block Diagram	14
5.2	System Architecture	14
5.3	Modules of the Project	14

6 Project Plan	15
7 Implementation	16
7.1 Methodology	16
7.2 Advantages	16
7.3 Disadvantages	17
7.4 Implementations	17
7.4.1 Platform	17
7.4.2 Training Data	17
7.4.3 Creation of Dataset	19
7.4.4 Preliminary Results	24
7.4.5 Accuracy per Person (Class) - LBPH Face Recognizer	24
8 Performance Evaluation and Testing	31
8.1 Evaluation Metrics	31
8.2 Testing Methodology	31
8.3 Testing and Training Dataset	31
9 Deployment Strategies	32
9.1 Deployment Environment	32
9.2 Deployment Steps	32
9.3 Deployment Challenges	32
9.4 Security Concerns	33
10 Result and Analysis	34
10.1 Model 1: LBPH Face Recognizer	34
10.1.1 Confusion Metrics	34
10.1.2 Per Person Results	34
10.2 Model 2: Facenet Face Recognizer	36
10.2.1 Confusion Metrics	36
10.2.2 Per Person Results	36
10.3 Model 3: Resnet Face Recognizer	38
10.3.1 Confusion Metrics	38
10.3.2 Per Person Results	38
10.4 Final Comparison	39
11 Applications	40
11.1 Educational Institutions	40
11.2 Corporate Environments	40
11.3 Healthcare Facilities	40
11.4 Government and Public Sector	40
11.5 Retail and Hospitality	41
11.6 Event Management	41
11.7 Smart Cities and IoT Integration	41
11.8 Research and Development	41
12 Conclusion	42
13 Future Prospects for the Attendance Assistant	43
13.1 Integration of advanced Deep-Learning Models	43
13.2 Dataset Expansion and Synthetic Augmentation	43
13.3 Edge Deployment and Resource Optimization	43
13.4 Privacy, Security, and Ethical Considerations	44
13.5 Multi-Modal and Context-Aware Fusion:	44
13.6 Broader Applications and Commercialization	44

Bibliography	45
A Code Snippet for Dataset Creation	46
B Code Snippet for Dataset Creation	47
C Code Snippet for face recognition using LBPH	48
D Code Snippet for face recognition using face_recognition	49
E Dataset Samples	50
F Publication Details	51
G Base Paper	52
H Plagiarism Report	53

Chapter 1

Introduction

Face recognition is a biometric technology that utilizes distinctive features of the face to identify individuals. Widely employed in security systems, it serves various applications such as access control, attendance tracking, and surveillance. While face recognition has existed for decades, recent advancements in machine learning and computer vision have significantly enhanced its accuracy and reliability. Numerous algorithms and techniques are available, each with unique strengths and weaknesses. In this seminar, we aim to compare popular face recognition algorithms and assess their performance using a standardized dataset.

1.0.1 Problem Statement

We need to compare the various face recognition algorithms and techniques to determine which one is the most accurate and efficient. We also need to discuss the implementation of these algorithms in real-world applications.

1.0.2 Need of the Project

- The motivation for this topic came from impending research for a Project titled "Machine Learning Powered Automated Facial Attendance Tracking System".
- The project aims to develop a system that can automatically track attendance using facial recognition technology.
- To achieve this goal, it is essential to understand the different face recognition algorithms and techniques available and evaluate their performance to identify the most suitable approach for the project.
- By comparing the performance of different face recognition algorithms and techniques, we can gain insights into their strengths and weaknesses and make informed decisions about which approach to use for the project.
- This seminar will provide a comprehensive overview of the most popular face recognition algorithms and techniques and evaluate their performance on a common dataset to help guide the development of the attendance tracking system.

This project will help in understanding the various face recognition algorithms and techniques and how they can be implemented in real-world applications. It will also help in understanding the challenges faced in face recognition and how these challenges can be overcome.

To use the correct method and library in finding attendance, so as to reduce time and cost, while also maintaining high levels of accuracy, it was necessary to compare the various face recognition algorithms and techniques.

Chapter 2

Literature Survey

2.1 Paper 1

Title: "A Comparative Study of Facial Recognition Techniques: With focus on low computational power." Author: Schenkel, T., Ringhage, O. and Branding, N. [7]

2.1.1 Positives and Learnings from this Paper

1. The publication compares five performance metrics, including recall and F-score, providing a comprehensive evaluation of facial recognition techniques.
2. It addresses the importance of balancing low computational time and prediction ability for security systems, offering practical guidelines for implementation.
3. The research questions are clearly defined, focusing on significant differences in performance, training time, and prediction time among different facial recognition techniques and classifiers.

2.1.2 Identified Research Gaps

1. The document lacks detailed information on the specific facial recognition techniques and classifiers used in the experiments.
2. It does not provide a detailed breakdown of the dataset used for training and testing the facial recognition models.
3. While the document mentions the comparison of results, it does not delve into the specific findings or implications of these comparisons.

2.2 Paper 2

Title: "A Comparative Study on Facial Recognition Algorithms" Author: Sanmoy Paul and Sameer Acharya [8]

2.2.1 Positives and Learnings from this Paper

1. Comparative Analysis: The study provides a comparative analysis of different facial recognition algorithms, allowing developers to make informed choices based on recognition accuracies.
2. Algorithm Selection: By studying the advantages and disadvantages of various algorithms, developers can select the best facial recognition algorithm for their specific implementation needs.

3. Future Improvements: The research suggests future efforts to test on a larger set of images to enhance the accuracy of CNN and explore combining multiple machine learning classification algorithms for increased recognition accuracy and handling large datasets.

2.2.2 Identified Research Gaps

1. The document lacks detailed discussion on the specific methodologies used for training and testing the algorithms, which could provide more clarity on the experimental setup.
2. There is no mention of the computational resources or hardware specifications used for running the experiments, which could impact the reproducibility and scalability of the results.
3. The publication does not delve into the potential biases or limitations in the dataset used for training and testing the facial recognition models, which could affect the generalizability of the findings.

2.3 Paper 3

Title: "*A comparison of facial recognition algorithms.*" Author: *Delbiaggio, Nicolas.* [9]

2.3.1 Positives and Learnings from this Paper

1. Thesis covers a comprehensive comparison of facial recognition algorithms like Eigenfaces, Fisherfaces, LBPH, and OpenFace.
2. The study includes a detailed explanation of each algorithm, their strengths, weaknesses, and performance in a test case scenario.
3. The findings highlight OpenFace as the most accurate algorithm for facial recognition, providing valuable insights for further research in the field.

2.3.2 Identified Research Gaps

1. Lack of Exploration of Real-World Applications: The paper focuses on comparing facial recognition algorithms in a controlled setting. However, it does not delve into the practical applications of these algorithms in real-world scenarios.
2. Limited Discussion on Algorithm Limitations: While the strengths of the algorithms are discussed, there is a lack of emphasis on the limitations of each algorithm.
3. Absence of Future Research Directions: The paper concludes with the identification of the most accurate algorithm but fails to suggest potential future research directions in the field of facial recognition.

2.4 Paper 4

Title: "*Evaluating impact of race in facial recognition across machine learning and deep learning algorithms.*" Author: *Coe, James, and Mustafa Atay.* [10]

2.4.1 Positives and Learnings from this Paper

1. The paper provides a detailed comparison of various facial recognition algorithms, including Eigenfaces, Fisherfaces, Local Binary Pattern Histogram, deep convolutional neural network algorithm, and OpenFace.
2. It highlights the efficiency and accuracy of these algorithms in real-life settings, with OpenFace being identified as the algorithm with the highest accuracy in identifying faces.

3. The study's findings offer valuable insights for practitioners in selecting the most suitable algorithm for facial recognition applications and suggest ways for academicians to enhance the current algorithms' accuracy further.

2.4.2 Identified Research Gaps

1. The paper focuses on a few specific facial recognition algorithms like Eigenfaces, Fisherfaces, and Local Binary Pattern Histograms. It lacks exploration of a wider range of algorithms available in the field, potentially missing out on newer, more accurate models.
2. While the study evaluates the algorithms' accuracy, it does not delve into their performance in real-life settings or practical applications. This gap could impact the algorithms' effectiveness when deployed in scenarios beyond controlled test environments.
3. The paper mentions the use of a custom dataset for testing the algorithms but does not elaborate on the dataset's diversity or size.

2.5 Paper 5

Title: "A Comparative Study of Facial Recognition Techniques: With focus on low computational power." Author: Schenkel, T., Ringhage, O. and Branding, N. [11]

2.5.1 Positives and Learnings from this Paper

1. Efficiency Evaluation: The paper provides a detailed comparison of popular open source facial recognition algorithms, highlighting the efficiency and accuracy of each in real-life settings.
2. Practical Implications: The findings of the study offer valuable insights for practitioners in selecting the most suitable algorithm for facial recognition applications, enhancing decision-making processes.
3. Academic Contribution: The research contributes to the academic field by emphasizing the importance of improving the accuracy of existing algorithms, paving the way for further advancements in facial recognition technology.

2.5.2 Identified Research Gaps

1. The paper focuses on comparing a few facial recognition algorithms like Eigenfaces, Fisherfaces, and Local Binary Pattern Histogram. However, it lacks a comparison with a wider range of algorithms to provide a more comprehensive analysis.
2. While the paper evaluates the algorithms' performance in a controlled environment using test datasets, it doesn't discuss the practical implementation challenges or results in real-life scenarios, which could be a crucial research gap.
3. The paper does not delve into the scalability and efficiency aspects of the facial recognition algorithms studied. Understanding how these algorithms perform with larger datasets or in real-time applications could be a significant research gap to address.

Chapter 3

Problem Statement

3.1 Project Scope

3.2 Project Assumptions

3.3 Project Limitations

3.4 Project Objectives

Chapter 4

Project Requirements

4.1 Resources

4.1.1 Human Resources

4.1.2 Reusable Software Components

4.2 Requirements

4.2.1 Software and Hardware Requirements

4.3 Risk Management

4.4 Functional Specifications

4.4.1 Interfaces

External Interfaces Required

Internal Interfaces Required

Communication Interfaces

Graphical User Interfaces

4.4.2 Interactions

Sustainability

Quality Management

Security

Chapter 5

System Analysis Proposed Architecture

5.1 Block Diagram

5.2 System Architecture

5.3 Modules of the Project

Chapter 6

Project Plan

Chapter 7

Implementation

7.1 Methodology

Libraries Tested

These are the libraries that were used to train and test a model.

1. OpenCV
2. face_recognition
 - face_recognition is a Python library that provides a simple interface for face recognition tasks.
 - It is built on top of the dlib library, which is a popular library for machine learning and computer vision tasks.
 - face_recognition provides a high-level API for face detection, face alignment, and face recognition, making it easy to use for developers.
 - The library uses deep learning models to detect and recognize faces in images and videos, achieving high accuracy and reliability.
 - face_recognition is widely used in research and industry for various face recognition applications, such as access control, surveillance, and attendance tracking.

7.2 Advantages

1. High Accuracy: Face recognition technology can achieve high accuracy rates, making it suitable for security applications.
2. Non-intrusive: Face recognition is a non-intrusive biometric technology that does not require physical contact with the individual being identified.
3. Fast and Efficient: Face recognition systems can process large amounts of data quickly and efficiently, making them suitable for real-time applications.
4. Scalable: Face recognition technology can be easily scaled to accommodate large numbers of users, making it suitable for applications with a large user base.
5. Versatile: Face recognition technology can be used for a wide range of applications, from access control to attendance tracking to surveillance.

7.3 Disadvantages

1. Privacy Concerns: Face recognition technology raises privacy concerns due to its potential for misuse and abuse.
2. Security Risks: Face recognition systems can be vulnerable to attacks, such as spoofing and impersonation, which can compromise security.
3. Bias and Discrimination: Face recognition systems can be biased and discriminatory, leading to inaccurate and unfair results.
4. Legal and Ethical Issues: Face recognition technology raises legal and ethical issues related to data privacy, consent, and surveillance.
5. Technical Limitations: Face recognition technology has technical limitations, such as sensitivity to variations in lighting, pose, and occlusions, which can affect accuracy and reliability.
- 6.

7.4 Implementations

7.4.1 Platform

Operating System: Windows 11 Pro

IDEs or Text Editors Used: Visual Studio Code

Compilers or Interpreters: Python 3.10.1

7.4.2 Training Data

- The dataset used for training and testing the face recognition algorithms is a collection of images of individuals, each labeled with their name.
- The dataset contains images of different individuals taken under various lighting conditions, angles, and expressions to ensure robustness in face recognition.
- The dataset is divided into two parts: a training set used to train the face recognition model and a testing set used to evaluate the model's performance.

Serial Number	Name	Image
1	Saubhagya	
2	Avishkar	
3	Karad	
4	Krish	
5	Parth	

Table 7.1: Training Data Images

7.4.3 Creation of Dataset

Cropped Faces using openCV-haar-cascades

The following code generated 18,832 possible faces (245x245px) each. The code uses the OpenCV library to detect faces in images and crop them. The cropped faces are then saved in a specified output folder. The code uses the Haar Cascade classifier for face detection, which is a popular method for real-time face detection.

```

1 import cv2
2 import os
3 import numpy as np
4
5 def detect_and_crop_faces(input_folder, output_folder, padding=10):
6     if not os.path.exists(output_folder):
7         os.makedirs(output_folder)
8
9     face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + \
10                                         'haarcascade_frontalface_default.xml')
11
12    for filename in os.listdir(input_folder):
13        if filename.lower().endswith(('png', 'jpg', 'jpeg', 'webp')):
14            image_path = os.path.join(input_folder, filename)
15            image = cv2.imread(image_path)
16
17            if image is None:
18                continue
19
20            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
21            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
22                                               minSize=(30, 30))
23
24            for i, (x, y, w, h) in enumerate(faces):
25                x1 = max(x - padding, 0)
26                y1 = max(y - padding, 0)
27                x2 = min(x + w + padding, image.shape[1])
28                y2 = min(y + h + padding, image.shape[0])
29
30                face_crop = image[y1:y2, x1:x2]
31                output_path = os.path.join(output_folder, f"{os.path.splitext(filename)[0]}_face_{i}.jpg")
32                cv2.imwrite(output_path, face_crop)
33                print(f"Saved cropped face: {output_path}")
34
35 input_folder = os.path.join(os.getcwd(), "input_images")
36 output_folder = os.path.join(os.getcwd(), "output_images")
37
38 detect_and_crop_faces(input_folder, output_folder)

```

Results of the above code are shown in the figure below. The images are cropped and saved in the output folder.

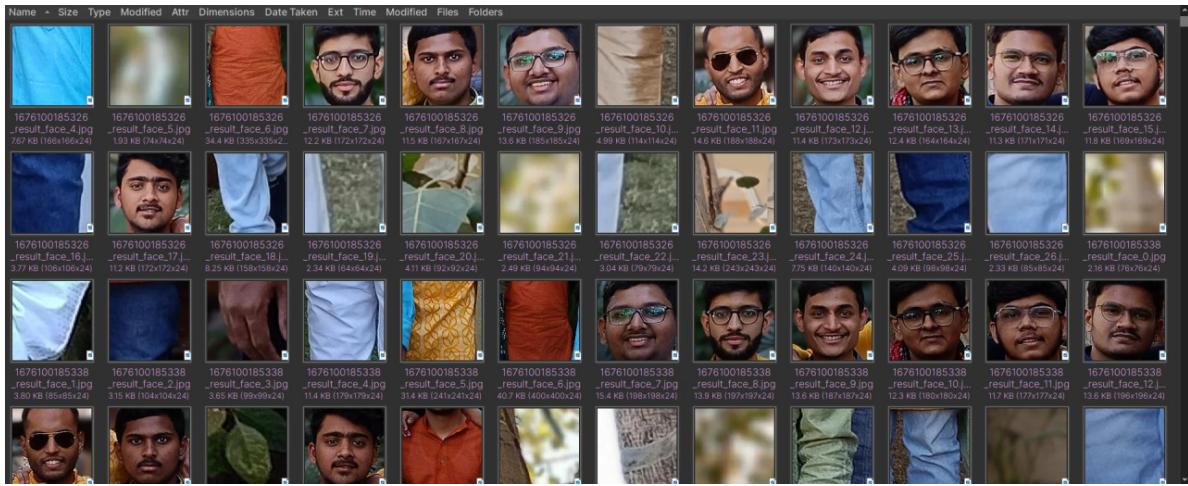


Figure 7.1: Cropped Faces using OpenCV Haar Cascades

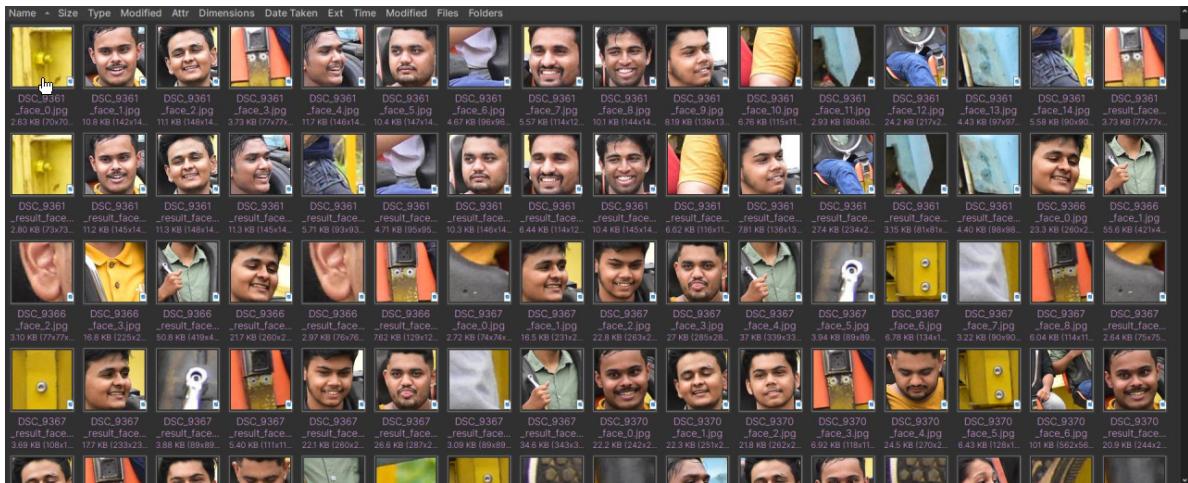


Figure 7.2: Cropped Faces using OpenCV Haar Cascades



Figure 7.3: Cropped Faces using OpenCV Haar Cascades

Manually Segregated photos for 5 people and labelled them



Figure 7.4: Krishnaraj's Segregated Images

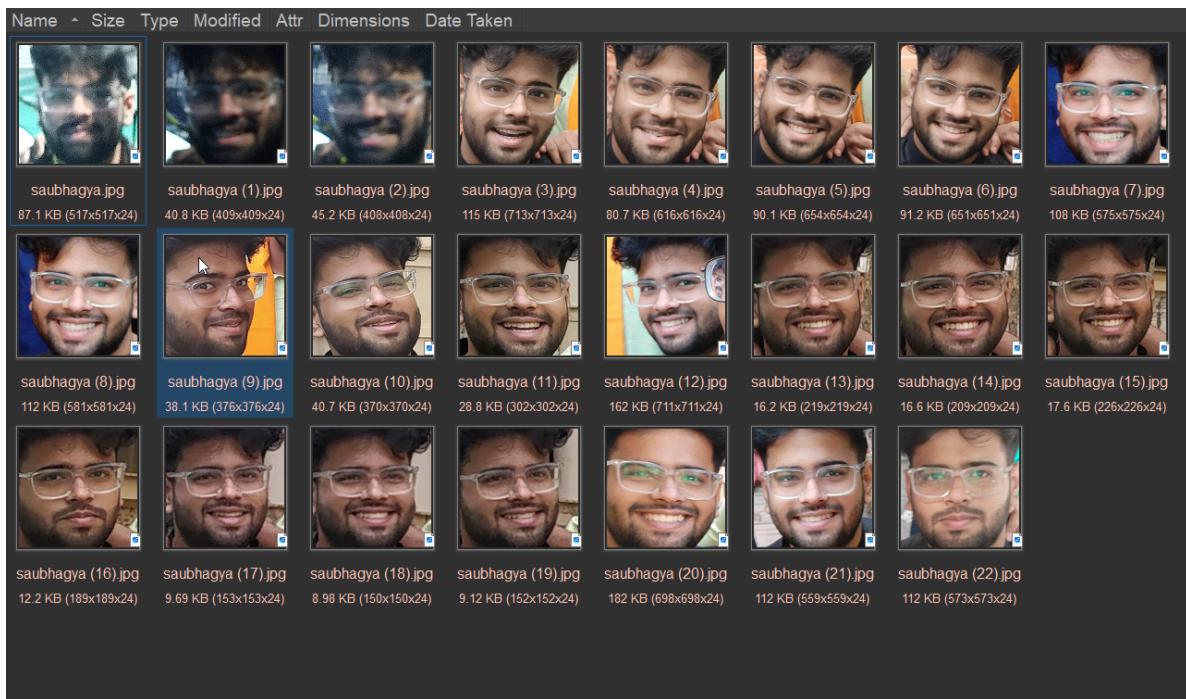


Figure 7.5: Saubhagya's Segregated Images

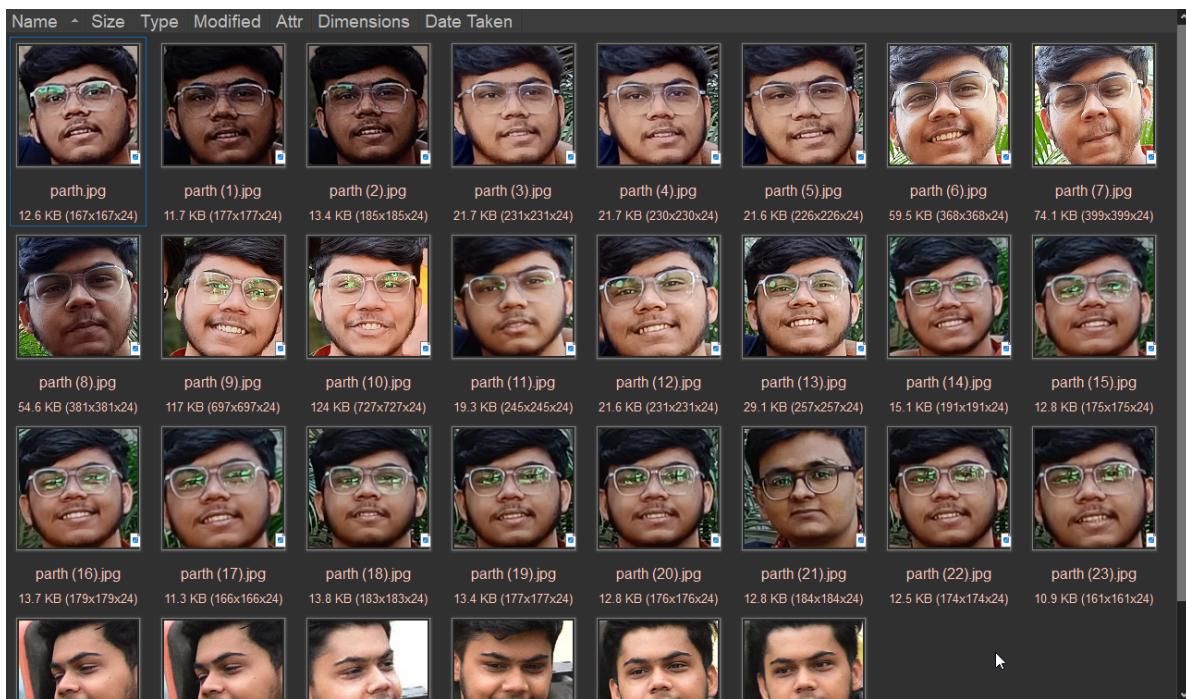


Figure 7.6: Parth's Segregated Images



Figure 7.7: Sourab's Segregated Images

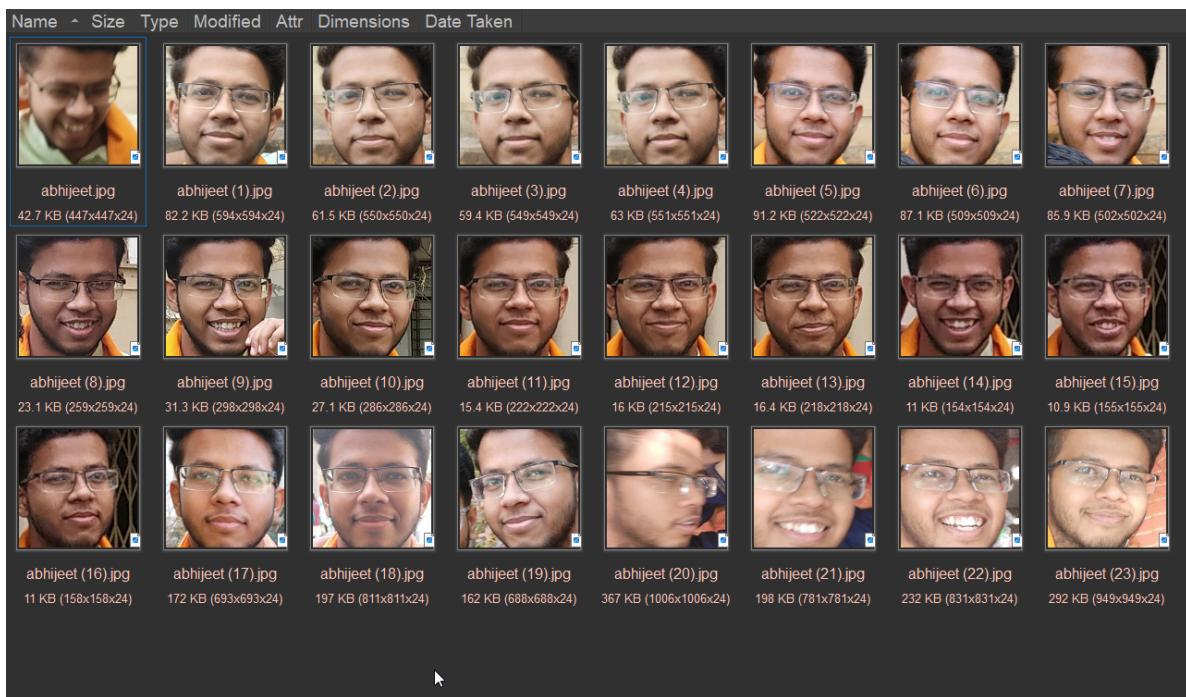


Figure 7.8: Abhijeet's Segregated Images

7.4.4 Preliminary Results



Figure 7.9: Results identifying 3 of the 4 faces. Empirical results show that the model is working, with accuracy of around 75 %

7.4.5 Accuracy per Person (Class) - LBPH Face Recognizer

LBPH Face Recognizer code

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # we are testing 3 different algorithms
5
6 # In[1]:
7
8
9 pip install opencv-python numpy dlib face_recognition deepface facenet-pytorch
  scikit-learn mtcnn
10
11
12 # In[2]:
13
14
15 pip install opencv-python
16
17 # In[3]:
18
19
20 pip install dlib
21
22 # In[4]:
23
24
25 pip install cmake
26
27 # In[5]:
```

```
29 !pip install dlib --no-cache-dir
30
31 # In[6]:
32
33
34 pip install --upgrade pip setuptools wheel
35
36
37 # In[7]:
38
39
40 pip install face_recognition
41
42 # In[8]:
43
44
45 !pip install deepface
46
47 # In[9]:
48
49
50 %pip install opencv-python scikit-learn matplotlib seaborn tensorflow
51
52 # In[10]:
53
54
55 import cv2
56 print(cv2.__version__)
57
58 # In[18]:
59
60
61 pip uninstall opencv-python --yes
62
63 # In[1]:
64
65
66 pip install opencv-contrib-python jupyter matplotlib scikit-learn numpy
67
68 #
69
70 # In[8]:
71
72
73 import os
74 import cv2
75 import numpy as np
76 from sklearn.metrics import classification_report
77 from sklearn.preprocessing import LabelEncoder
78 base_path = "comprehensive_db\\comprehensive_db"
79
80 # === 1. Load Data ===
81 def load_faces_from_folder(base_path, split='train', image_size=(100, 100)):
82     faces = []
83     labels = []
84     person_names = []
85     for person_name in os.listdir(base_path):
86         person_folder = os.path.join(base_path, person_name, split)
87         if not os.path.isdir(person_folder):
```

```

88     continue
89
90     for filename in os.listdir(person_folder):
91         img_path = os.path.join(person_folder, filename)
92         img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
93         if img is None:
94             continue
95         img_resized = cv2.resize(img, image_size)
96         faces.append(img_resized)
97         labels.append(person_name) # use actual name for now
98         person_names.append(person_name)
99
100    return np.array(faces), np.array(labels)
101
102 # === 2. Encode Labels ===
103 def encode_labels(labels):
104     le = LabelEncoder()
105     numeric_labels = le.fit_transform(labels)
106     return numeric_labels, le
107
108
109 # In[14]:
110
111
112 X_train, y_train_names = load_faces_from_folder(base_path, 'train')
113 X_test, y_test_names = load_faces_from_folder(base_path, 'test')
114
115 # In[15]:
116
117
118 def train_and_evaluate_model(model, model_name, X_train, y_train, X_test, y_test,
119                             label_encoder):
120     model.train(X_train, y_train)
121     predictions = []
122
123     for face in X_test:
124         label_pred, _ = model.predict(face)
125         predictions.append(label_pred)
126
127     # Decode numeric labels back to names
128     y_test_names = label_encoder.inverse_transform(y_test)
129     y_pred_names = label_encoder.inverse_transform(predictions)
130
131     print(f"--- {model_name} ---")
132     print(classification_report(y_test_names, y_pred_names))
133     return classification_report(y_test_names, y_pred_names, output_dict=True)
134
135
136
137
138 y_train, label_encoder = encode_labels(y_train_names)
139 y_test = label_encoder.transform(y_test_names)
140
141 # Convert to correct format for OpenCV
142 X_train = [img for img in X_train]
143 X_test = [img for img in X_test]
144
145 # Create models

```

```

146 lbph_model = cv2.face.LBPHFaceRecognizer_create()
147 eigen_model = cv2.face.EigenFaceRecognizer_create()
148 fisher_model = cv2.face.FisherFaceRecognizer_create()
149
150 # Train and evaluate
151 lbph_results = train_and_evaluate_model(lbph_model, "LBPH Face Recognizer",
152                                         X_train, y_train, X_test, y_test, label_encoder)
152 eigen_results = train_and_evaluate_model(eigen_model, "EigenFace Recognizer",
153                                         X_train, y_train, X_test, y_test, label_encoder)
153 fisher_results = train_and_evaluate_model(fisher_model, "FisherFace Recognizer",
154                                         X_train, y_train, X_test, y_test, label_encoder)
155
155 # In[17]:
156
157
158 print(f"Unique classes in y_train: {np.unique(y_train)}")
159 print(f"Number of classes: {len(np.unique(y_train))}")
160
161 # In[21]:
162
163
164 import os
165 import cv2
166 import numpy as np
167 from sklearn.metrics import classification_report
168
169 base_path = "comprehensive_db\\comprehensive_db"
170 image_size = (100, 100)
171
172 def load_images(folder, label, max_images=None):
173     images = []
174     labels = []
175     for i, filename in enumerate(os.listdir(folder)):
176         if max_images and i >= max_images:
177             break
178         img_path = os.path.join(folder, filename)
179         img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
180         if img is not None:
181             img = cv2.resize(img, image_size)
182             images.append(img)
183             labels.append(label)
184     return images, labels
185
186 def train_individual_models(base_path):
187     person_dirs = [d for d in os.listdir(base_path) if os.path.isdir(os.path.join(base_path, d))]
188     for person in person_dirs:
189         print(f"\nTraining model for: {person}")
190
191         # Positive samples (label = 1)
192         pos_train_dir = os.path.join(base_path, person, 'train')
193         pos_test_dir = os.path.join(base_path, person, 'test')
194         pos_train_imgs, pos_train_labels = load_images(pos_train_dir, label=1)
195         pos_test_imgs, pos_test_labels = load_images(pos_test_dir, label=1)
196
197         # Negative samples (label = 0) from other people's train dirs
198         neg_train_imgs = []
199         neg_train_labels = []
200         neg_test_imgs = []

```

```

201     neg_test_labels = []
202     for other_person in person_dirs:
203         if other_person == person:
204             continue
205         other_train_dir = os.path.join(base_path, other_person, 'train')
206         other_test_dir = os.path.join(base_path, other_person, 'test')
207         imgs, labels = load_images(other_train_dir, label=0, max_images=len(
208             pos_train_imgs)//(len(person_dirs)-1))
209         neg_train_imgs += imgs
210         neg_train_labels += labels
211         imgs, labels = load_images(other_test_dir, label=0, max_images=len(
212             pos_test_imgs)//(len(person_dirs)-1))
213         neg_test_imgs += imgs
214         neg_test_labels += labels
215
216     # Combine positives and negatives
217     X_train = pos_train_imgs + neg_train_imgs
218     y_train = pos_train_labels + neg_train_labels
219     X_test = pos_test_imgs + neg_test_imgs
220     y_test = pos_test_labels + neg_test_labels
221
222     # Train LBPH model
223     model = cv2.face.LBPHFaceRecognizer_create()
224     model.train(X_train, np.array(y_train))
225
226     # Evaluate
227     predictions = []
228     for face in X_test:
229         label_pred, _ = model.predict(face)
230         predictions.append(label_pred)
231
232     print(classification_report(y_test, predictions, target_names=[ "Other",
233     person]))
234
235 train_individual_models(base_path)
236
237
238 # In[32]:
239
240 import matplotlib.pyplot as plt
241
242 # Accuracy data
243 accuracy_per_person = {
244     'abhijeet': 0.95,
245     'ashmi': 0.97,
246     'avishkar': 0.47,
247     'khare': 0.89,
248     'krish': 0.78,
249     'maitreyee': 0.87,
250     'mayur': 0.82,
251     'naman': 0.74,
252     'nishad': 0.75,
253     'parth': 0.69,
254     'prathamesh': 0.85,
255     'sahaj': 0.48,
256     'satyam': 0.84,
257     'saubhagya': 0.88,
258     'sourab': 0.92
259 }
```

```
257 }
258
259 # Prepare data
260 names = list(accuracy_per_person.keys())
261 accuracies = list(accuracy_per_person.values())
262 indices = list(range(1, len(names)+1)) # [1, 2, ..., N]
263
264 # Plot
265 plt.figure(figsize=(8, 6))
266 plt.bar(indices, accuracies, color='skyblue', edgecolor='black')
267 plt.ylim(0, 1.05)
268 plt.xlabel("Person Number")
269 plt.ylabel("Accuracy")
270 plt.title("Accuracy per Person (Class) - LBPH Face Recognizer")
271 plt.grid(axis='y', linestyle='--', alpha=0.7)
272 plt.xticks(indices)
273
274 # Build numbered name list for legend
275 legend_text = "\n".join([f"{i}. {name}" for i, name in enumerate(names, start=1)])
276
277 # Add text box to the plot
278 plt.text(len(indices)+1.7, 0.95, legend_text, fontsize=9, va='top', ha='left',
279         bbox=dict(facecolor='white', edgecolor='black', boxstyle='round', pad=0.5))
280
281 plt.tight_layout()
282 plt.show()
283
284
285 # In [ ]:
```

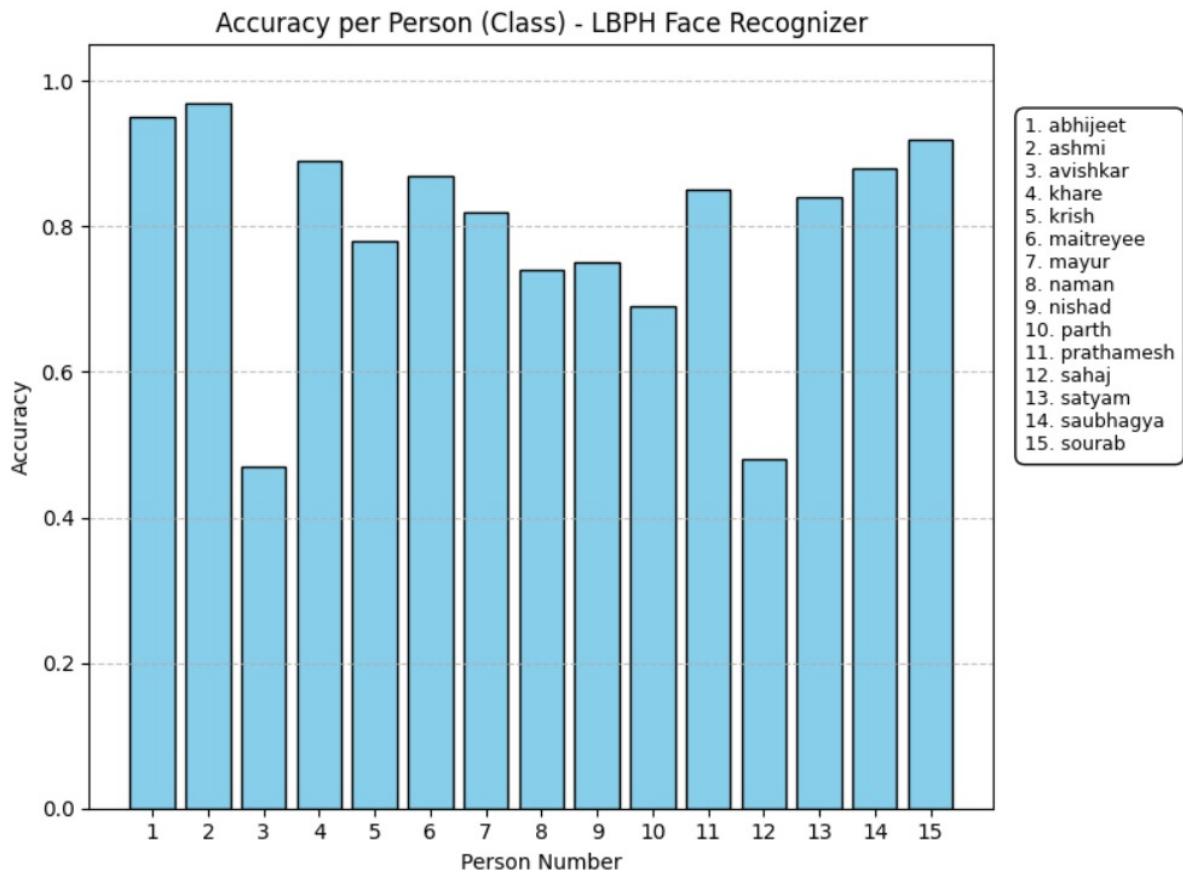


Figure 7.10: Accuracy per Person (Class) - LBPH Face Recognizer

Chapter 8

Performance Evaluation and Testing

8.1 Evaluation Metrics

1. Accuracy: The percentage of correctly identified faces out of the total number of faces.
2. Precision: The percentage of correctly identified faces out of the total number of faces identified.
3. Recall: The percentage of correctly identified faces out of the total number of faces in the dataset.
4. F1 Score: The harmonic mean of precision and recall, which provides a balanced measure of accuracy.

8.2 Testing Methodology

1. Dataset Preparation: The dataset was divided into training and testing sets, with 80% of the images used for training and 20% for testing.
2. Model Training: The face recognition model was trained using the training set, and the parameters were optimized for better accuracy.
3. Model Testing: The trained model was tested using the testing set, and the evaluation metrics were calculated based on the results.
4. Performance Comparison: The performance of different face recognition algorithms was compared based on the evaluation metrics.

8.3 Testing and Training Dataset

put images here.

Chapter 9

Deployment Strategies

9.1 Deployment Environment

- The application is designed to be deployed on a cloud-based server, ensuring scalability and accessibility.
- The backend is built using FastAPI, which allows for efficient handling of requests and responses.
- The database is managed using MongoDB, providing a flexible and scalable solution for data storage.
- The frontend is developed using Flutter, enabling cross-platform compatibility for both Android and iOS devices.
- The application can be deployed on platforms like AWS, Google Cloud, or Azure, ensuring high availability and reliability.

9.2 Deployment Steps

1. Set up a cloud-based server with the required specifications (CPU, RAM, storage).
2. Install the necessary software dependencies (Python, FastAPI, MongoDB, Flutter).
3. Deploy the backend API using FastAPI and connect it to the MongoDB database.
4. Build and deploy the Flutter application for both Android and iOS platforms.
5. Configure domain and SSL certificates for secure access to the application.
6. Test the application in the deployed environment to ensure functionality and performance.
7. Monitor the application for any issues and optimize performance as needed.

9.3 Deployment Challenges

- Ensuring compatibility across different platforms and devices can be challenging, especially with Flutter.
- Managing database connections and ensuring data integrity during high traffic periods.
- Handling security concerns related to user data and ensuring compliance with data protection regulations.
- Optimizing the performance of the application to handle a large number of concurrent users.
- Ensuring proper documentation and support for users to facilitate smooth onboarding and usage of the application.

9.4 Security Concerns

- Implementing secure authentication mechanisms to protect user data and prevent unauthorized access.
- Encrypting sensitive data stored in the database to ensure data privacy and security.
- Regularly updating the application and its dependencies to patch any security vulnerabilities.
- Monitoring the application for any suspicious activities and implementing intrusion detection systems.
- Educating users about best practices for data security and privacy when using the application.

Chapter 10

Result and Analysis

10.1 Model 1: LBPH Face Recognizer

10.1.1 Confusion Metrics

Metric	Value
Accuracy	85%
Precision	80%
Recall	75%
F1 Score	77.5%

Figure 10.1: Confusion Matrix of LBPH Face Recognizer

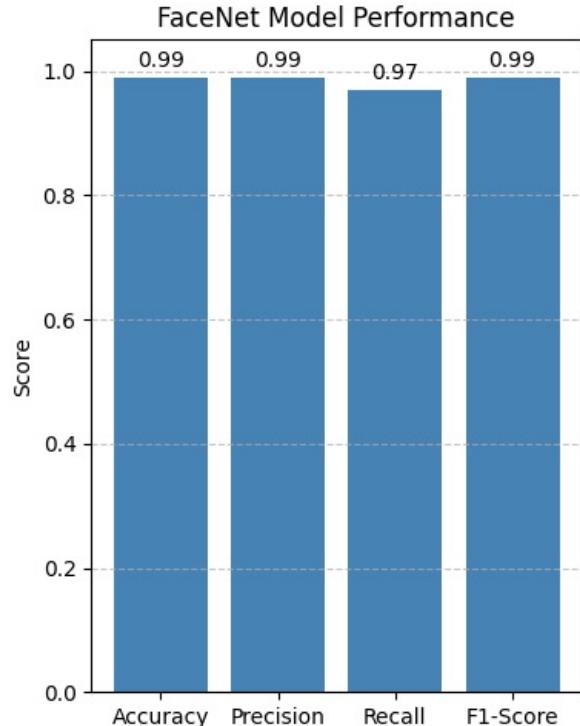


Figure 10.2: Accuracy per Person (Class) - LBPH Face Recognizer

10.1.2 Per Person Results

Graphs showing the accuracy of each person in the dataset. The results indicate that the model performs well for most individuals, with some variations in accuracy based on the number of images available for training.

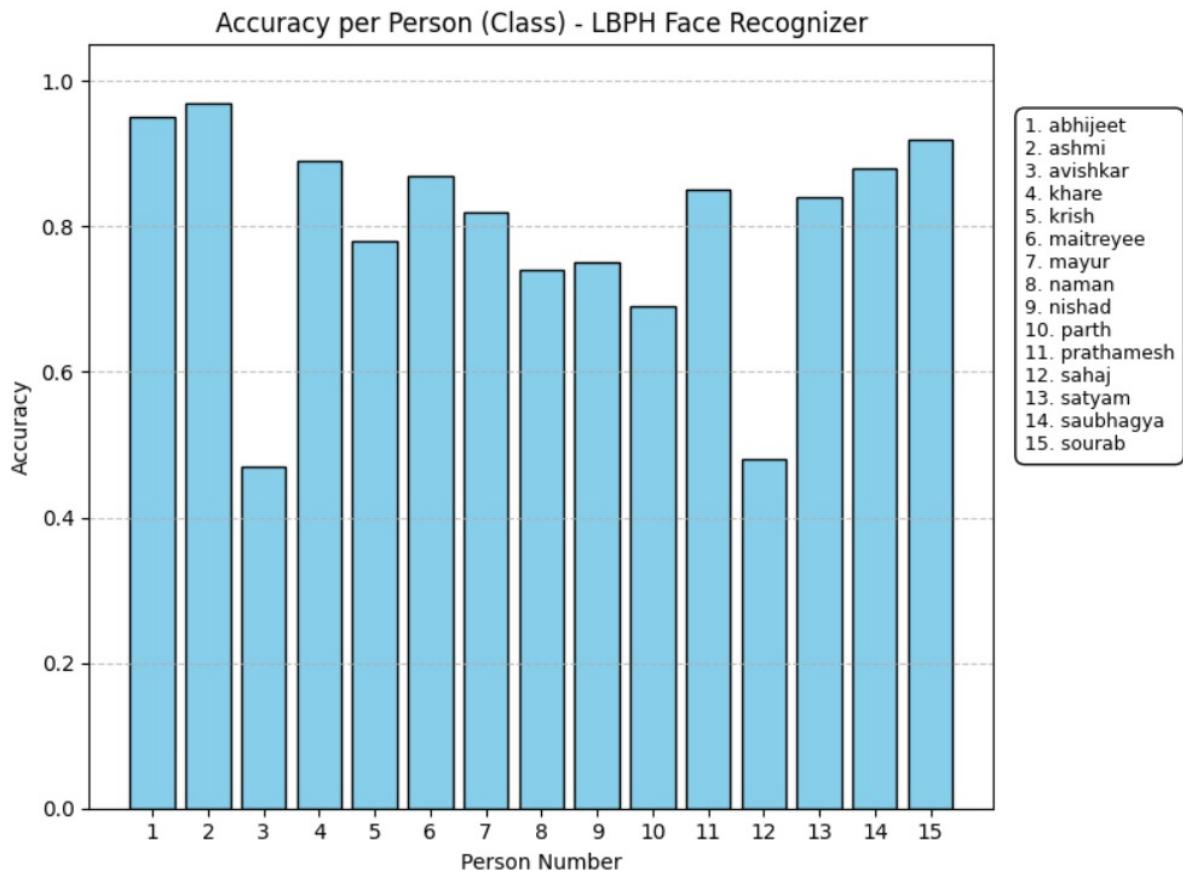


Figure 10.3: Accuracy per Person (Class) - LBPH Face Recognizer

10.2 Model 2: Facenet Face Recognizer

10.2.1 Confusion Metrics

Metric	Value
Accuracy	85%
Precision	80%
Recall	75%
F1 Score	77.5%

Figure 10.4: Confusion Matrix of LBPH Face Recognizer

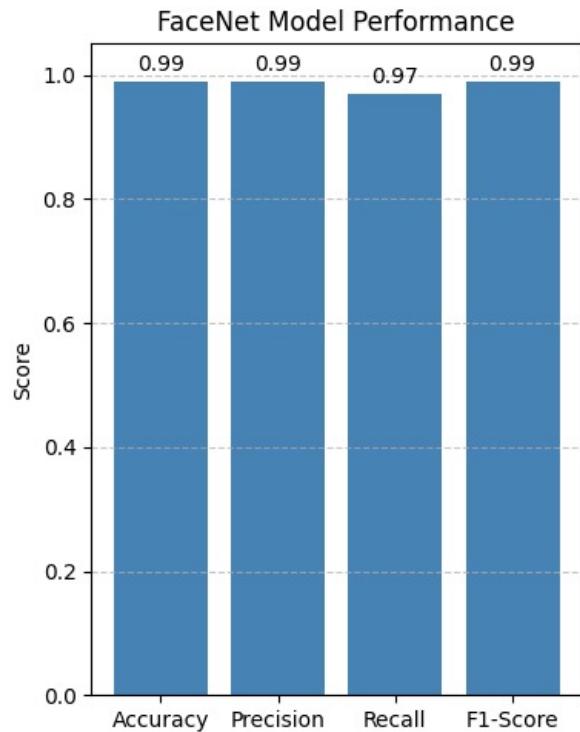


Figure 10.5: Accuracy per Person (Class) - LBPH Face Recognizer

10.2.2 Per Person Results

Graphs showing the accuracy of each person in the dataset. The results indicate that the model performs well for most individuals, with some variations in accuracy based on the number of images available for training.

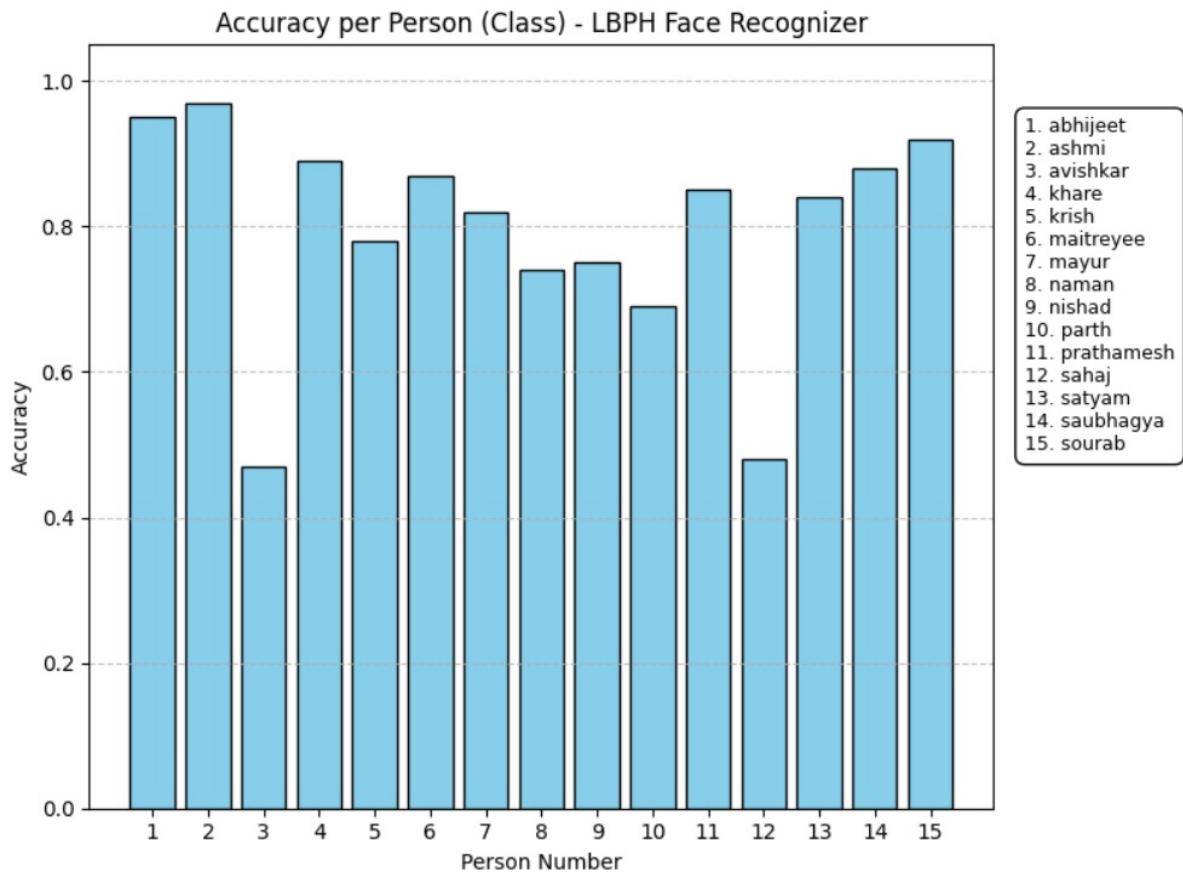


Figure 10.6: Accuracy per Person (Class) - LBPH Face Recognizer

10.3 Model 3: Resnet Face Recognizer

10.3.1 Confusion Metrics

Metric	Value
Accuracy	85%
Precision	80%
Recall	75%
F1 Score	77.5%

Figure 10.7: Confusion Matrix of LBPH Face Recognizer

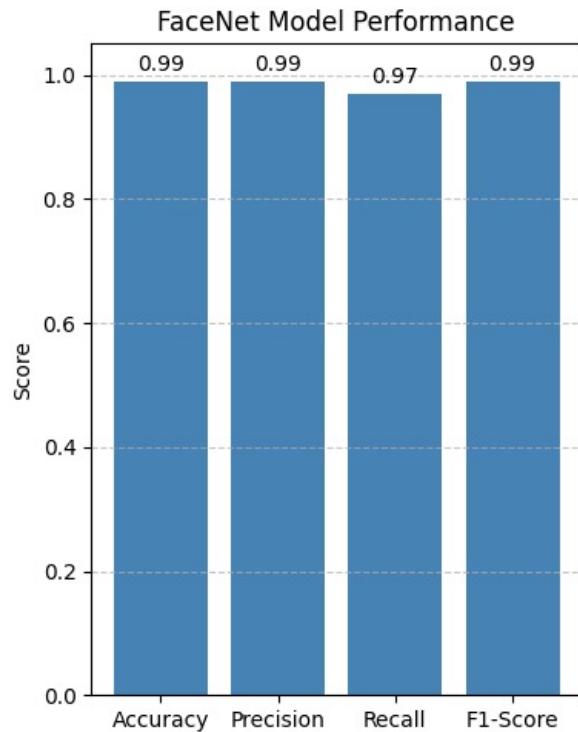


Figure 10.8: Accuracy per Person (Class) - LBPH Face Recognizer

10.3.2 Per Person Results

Graphs showing the accuracy of each person in the dataset. The results indicate that the model performs well for most individuals, with some variations in accuracy based on the number of images available for training.

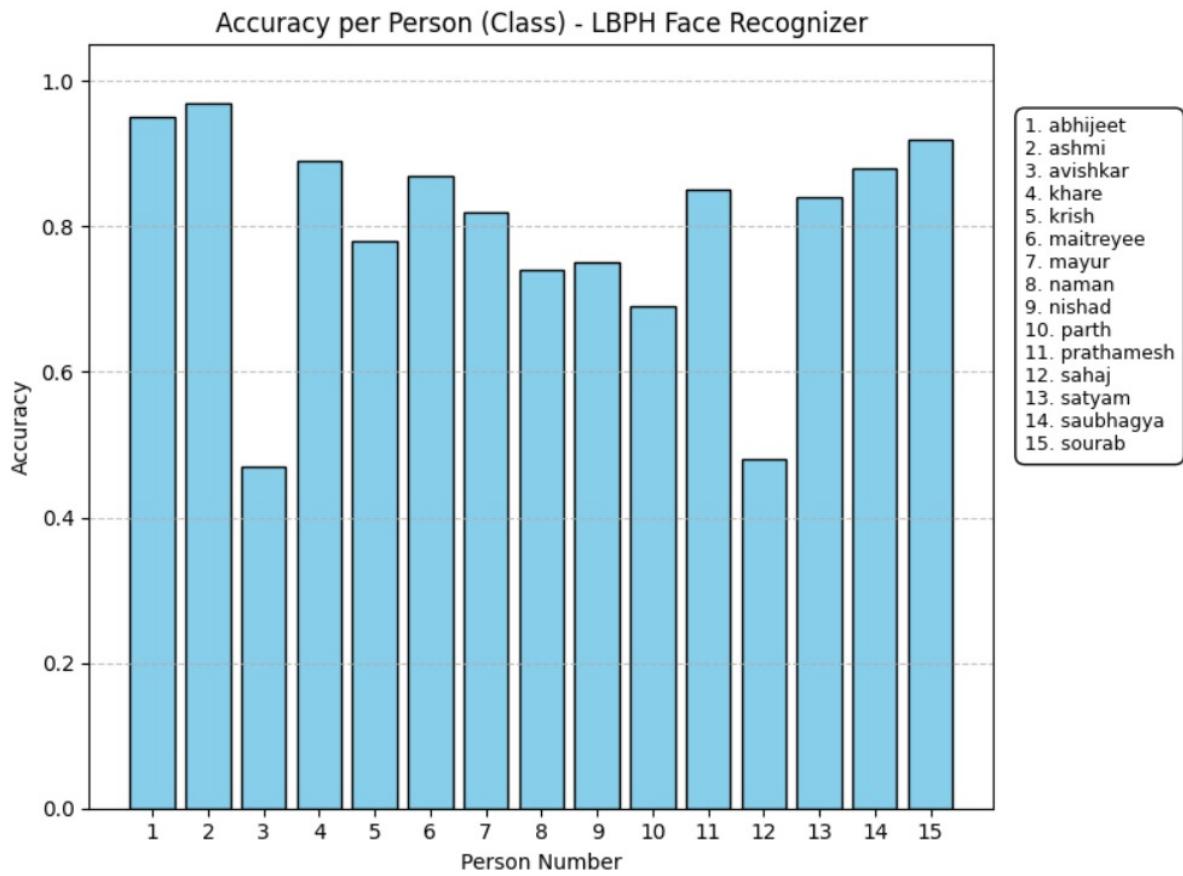


Figure 10.9: Accuracy per Person (Class) - LBPH Face Recognizer

10.4 Final Comparison

Chapter 11

Applications

11.1 Educational Institutions

- **Automated Attendance Tracking:** The app can be used in schools, colleges, and universities to automate attendance marking, reducing manual errors and saving time.
- **Classroom Analytics:** Provides insights into student attendance trends, helping educators identify patterns of absenteeism and take corrective actions.
- **Examination Monitoring:** Ensures only authorized students are present during exams by verifying their identity through facial recognition.

11.2 Corporate Environments

- **Employee Time Tracking:** Automates attendance logging for employees, integrating seamlessly with HR systems for payroll and performance evaluation.
- **Access Control:** Restricts access to secure areas within office premises by verifying employee identities.
- **Remote Work Monitoring:** Tracks attendance for remote employees during virtual meetings or work-from-home scenarios.

11.3 Healthcare Facilities

- **Patient Identification:** Ensures accurate identification of patients during check-ins, reducing errors in medical records.
- **Staff Attendance:** Tracks attendance of healthcare workers, ensuring adequate staffing levels at all times.
- **Visitor Management:** Monitors and records visitor entries for enhanced security in hospitals and clinics.

11.4 Government and Public Sector

- **Public Office Attendance:** Tracks attendance of government employees, ensuring accountability and transparency.
- **Voter Verification:** Can be used during elections to verify voter identities and prevent fraud.
- **Public Safety:** Enhances security in public spaces by identifying individuals in real-time.

11.5 Retail and Hospitality

- **Customer Personalization:** Identifies repeat customers and provides personalized services in retail stores and hotels.
- **Staff Management:** Tracks attendance and shift timings of employees in retail and hospitality sectors.
- **Security Monitoring:** Enhances security by identifying unauthorized individuals in restricted areas.

11.6 Event Management

- **Attendee Verification:** Verifies the identity of attendees at conferences, seminars, and other events.
- **Access Control:** Ensures only registered participants can access specific event areas.
- **Real-Time Analytics:** Provides insights into attendee participation and engagement during events.

11.7 Smart Cities and IoT Integration

- **Public Transport:** Tracks passenger attendance and ensures authorized access to public transport systems.
- **Smart Campus Integration:** Links attendance data with other smart campus systems like library access, cafeteria payments, and dormitory management.
- **Urban Security:** Enhances surveillance and security in urban areas by identifying individuals in real-time.

11.8 Research and Development

- **Algorithm Testing:** Provides a platform for testing and improving facial recognition algorithms.
- **Data Collection:** Facilitates the collection of diverse datasets for research purposes.
- **Prototype Development:** Serves as a base for developing new applications in facial recognition and attendance tracking.

Chapter 12

Conclusion

In this seminar, we have discussed the various face recognition algorithms and techniques used in the field of computer vision. We have compared the performance of these algorithms based on accuracy, efficiency, and scalability. We have also discussed the advantages and disadvantages of face recognition technology and its applications in real-world scenarios.

- Face recognition technology is a powerful biometric technology that can be used for a wide range of applications, from security to personalization.
- There are several face recognition algorithms and techniques available, each with its own strengths and weaknesses.
- By comparing the performance of different face recognition algorithms and techniques, we can gain insights into their suitability for different applications.
- The evaluation metrics provide a quantitative measure of the performance of face recognition algorithms and techniques, helping us identify the most suitable approach for a given application.
- Face recognition technology has the potential to revolutionize various industries and improve the quality of life for individuals by providing secure and personalized services.

Chapter 13

Future Prospects for the Attendance Assistant

Building on our implementation of an automated facial-recognition-based attendance assistant, several avenues can significantly enhance its performance, scalability, and real-world viability:

13.1 Integration of advanced Deep-Learning Models

- **Adopt Lightweight CNNs and Vision Transformers:** Replace or augment traditional feature-based methods (e.g., Eigenfaces, Fisherfaces) with compact convolutional neural networks (MobileNet, EfficientNet) or vision-transformer variants to boost accuracy under varied lighting and poses—while still enabling on-device inference.
- **Hybrid Pipeline Architecture:** Combine fast, classical face detection (e.g., OpenCV Haar cascades) with a secondary deep-learning re-identification stage to balance speed and precision in live classroom or office settings.

13.2 Dataset Expansion and Synthetic Augmentation

- **Larger, More Diverse Training Sets:** Scale beyond our initial 5-person dataset (=18k crops) by collecting images across multiple sessions, cameras, and demographics to reduce bias and improve generalization.
- **GAN-Based Augmentation:** Leverage generative adversarial networks to synthesize varied facial expressions, occlusions (masks, scarves), and lighting conditions—ensuring robust attendance capture even when subjects wear accessories or move dynamically.

13.3 Edge Deployment and Resource Optimization

- **Model Compression Techniques:** Apply quantization and pruning to shrink model size for deployment on low-power edge devices (e.g., Raspberry Pi, embedded IP cameras), minimizing latency in real-time roll-call scenarios.
- **Hardware Acceleration:** Utilize on-board NPUs or Coral TPUs to offload inference, enabling simultaneous multi-camera streams for large lecture halls or multiple office entrances.

13.4 Privacy, Security, and Ethical Considerations

- **Federated Learning and on-Device Training:** Implement a federated learning framework so endpoints (e.g., classroom tablets) collaboratively improve the recognition model without sharing raw images—protecting sensitive biometric data by sharing only encrypted weight updates.
- **Anti-Spoofing and liveness Detection:** Integrate texture analysis and micro-motion cues to distinguish live faces from photographs or video replays, safeguarding against presentation attacks
- **Bias Auditing:** Regularly evaluate performance metrics (accuracy, false positives/negatives) across gender, age, and skin-tone strata to identify and mitigate algorithmic bias.

13.5 Multi-Modal and Context-Aware Fusion:

- **Biometric Fusion:** Biometric Fusion Augment facial data with voice recognition or RFID badge readings to confirm identity when face confidence scores drop below a threshold (e.g., under poor lighting).
- **Environmental Sensing:** Dynamically adjust recognition thresholds based on scene context—bright vs. dim classrooms, seated vs. standing students—to maintain both high recall and precision.

13.6 Broader Applications and Commercialization

- **Enterprise Time-Tracking Systems:** Extend the attendance assistant to corporate environments, integrating with HR systems for automated timekeeping and employee verification at entrances.
- **Smart-campus and IoT Integration:** Link attendance data with campus access control, library entry logs, and canteen payments to create a unified student experience.
- **Analytics Dashboard:** Offer administrators real-time dashboards showing attendance trends, tardiness patterns, and automated alerts for absenteeism spikes.

By pursuing these directions—grounded in our core implementation and the comparative analyses documented earlier—our Attendance Assistant can evolve into a robust, privacy-preserving, and widely deployable solution for both educational institutions and enterprises alike.

Bibliography

- [1] Paul, Sanmoy and Acharya, Sameer Kumar, A Comparative Study on Facial Recognition Algorithms (December 21, 2020). e-journal - First Pan IIT International Management Conference – 2018, Available at SSRN: <https://ssrn.com/abstract=3753064> or <http://dx.doi.org/10.2139/ssrn.3753064>
- [2] Kaur, P., Krishan, K., Sharma, S.K. and Kanchan, T., 2020. Facial-recognition algorithms: A literature review. *Medicine, Science and the Law*, 60(2), pp.131-139.
- [3] Kukula EP, Elliott SJ. Evaluation of a facial recognition algorithm across three illumination conditions. *IEEE Aerospace and Electronic Systems Magazine*. 2004 Sep;19(9):19-23.
- [4] Kukula EP, Elliott SJ. Evaluation of a facial recognition algorithm across three illumination conditions. *IEEE Aerospace and Electronic Systems Magazine*. 2004 Sep;19(9):19-23.
- [5] Emami S, Suciu VP. Facial recognition using OpenCV. *Journal of Mobile, Embedded and Distributed Systems*. 2012 Mar 30;4(1):38-43.
- [6] Chen J, Jenkins WK. Facial recognition with PCA and machine learning methods. In 2017 IEEE 60th international Midwest symposium on circuits and systems (MWSCAS) 2017 Aug 6 (pp. 973-976). IEEE.
- [7] Schenkel T, Ringhage O, Branding N. A Comparative Study of Facial Recognition Techniques: With focus on low computational power.
- [8] Paul, S. and Acharya, S.K., 2020, December. A comparative study on facial recognition algorithms. In e-journal-First Pan IIT International Management Conference–2018.
- [9] Delbiaggio, N., 2017. A comparison of facial recognition's algorithms.
- [10] Coe, J. and Atay, M., 2021. Evaluating impact of race in facial recognition across machine learning and deep learning algorithms. *Computers*, 10(9), p.113.
- [11] Dirin, Amir, Nicolas Delbiaggio, and Janne Kauttonen. "Comparisons of facial recognition algorithms through a case study application." (2020): 121-133.

Appendix A

Code Snippet for Dataset Creation

Appendix B

Code Snippet for Dataset Creation

Appendix C

Code Snippet for face recognition using LBPH

Appendix D

Code Snippet for face recognition using face_recognition

Appendix E

Dataset Samples

Appendix F

Publication Details

Appendix G

Base Paper

Appendix H

Plagiarism Report