Mini Project Report

Entitled

Face Recognition using Machine Learning

Submitted to the Department of Electronics Engineering in Partial Fulfilment for the Requirements for the Degree of

Bachelor of Technology
(Electronics and Communication)

: Presented & Submitted By :

Jinhal Maheshwari, Shreeya Dave, Shakshi Singh Rajput

Roll No. (U21EC082, U21EC129, U21EC138)

B. TECH. VI (EC), 6^{th} Semester

: Guided By :

Dr. Kishor Upla
Assistant Professor, SVNIT

(Year: 2023-24)

DEPARTMENT OF ELECTRONICS ENGINEERING

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

Surat-395007, Gujarat, INDIA.

Sardar Vallabhbhai National Institute Of Technology

Surat - 395 007, Gujarat, India

DEPARTMENT OF ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project Report entitled "**Face Recognition using Machine Learning**" is presented & submitted by Jinhal Maheshwari, Shreeya Dave, Shakshi Singh Rajput, bearing Roll No. U21EC082, U21EC129, U21EC138, of B.Tech. VI, 6^{th} Semester in the partial fulfillment of the requirement for the award of B.Tech. Degree in Electronics & Communication Engineering for academic year 2023-24.

They have successfully and satisfactorily completed their **Mini-Project** in all respects. We, certify that the work is comprehensive, complete and fit for evaluation.

Dr. Kishor Upla
Assistant Professor & Project Guide

Abstract

This report presents a comprehensive exploration of face recognition algorithms, focusing on the implementation of two key methodologies: Local Binary Pattern Histogram (LBPH) and Support Vector Machines (SVM). Leveraging the OpenCV library and Raspberry Pi hardware, we conducted experiments to train and test these algorithms on a dataset comprising images of team members.

The LBPH algorithm offers a straightforward yet effective approach to facial feature extraction, while SVMs provide a robust method for pattern recognition, particularly in discriminating between faces and non-faces. Our experimental setup involved capturing facial images, training the recognition models, and evaluating their performance using metrics such as accuracy, precision, recall, and F1-score.

Furthermore, we investigated the hardware implementation of these algorithms on a Raspberry Pi, demonstrating the feasibility of deploying face recognition systems in embedded environments. Overall, this report contributes to the understanding of face recognition technology and its practical applications, showcasing its potential in various fields such as security, authentication, and personalized user experiences.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Overview

One of the several wonders that artificial intelligence technology has contributed to the world is facial recognition. To identify the category of every input object, face recognition systems compare face images with a collected group of images called dataset. Facial recognition analysis has become one of the interesting research areas for experts in human-computer interaction applications [1].

In reality, people's identification has been performed by utilizing facial features since the start of time. Face images are used throughout the globe to recognize persons with citizenship identification, identification cards, social security card, intrusion detection, etc.

This process includes segmentation, isolation, and validation of facial features from the unstable environment and likely real faces. The initial effort to identify face was done by calculating unique facial characteristics such as nose size, brows width, and forehead area. Face recognition was introduced as an authentication tool in the latest gadgets [2].Face recognition is widely used in security systems to identify and authenticate individuals. It is used in airports, border control, government facilities, and commercial buildings to enhance security measures. In schools, colleges, and workplaces, face recognition is used for automated attendance tracking. It eliminates the need for manual attendance marking and reduces errors.

1.2 Objectives

The process of face recognition comprises two major steps, the extraction of the feature and the classification. Raw face pictures can take a lot of time to identify as it results from an enormous amount of pixels. The number of pixels must be reduced. This is called reducing dimensional space or removal of features to save time for the phase of the assessment. The extraction of features corresponds to the conversion of face space into a space of feature [3].

Classification is the mechanism by which the class of variables is predicted. Occasionally classes are referred to as labels or levels. For example, a classifier is utilized

to categorize certain characteristics extracted from a face picture and to produce a label that is used to identify a person.

Chapter 2
Face Recognition Algorithms

2.1 LBPH

LBPH algorithm stands for Local Binary Pattern Histogram which is a simple solution for the face recognition problem [4] [5]. It can be distinguishing both the front face, and side face. LBPH algorithm is a simple and efficient text description operator. The picture pixels are labeled using threshold value of each pixel and produce a binary number [6]. Then the Local Binary Pattern Histogram is combined with histogram and characterizes the face images with a simple data vector.

2.1.1 The LBPH Face Recognizer Process

The Local Binary Patterns Histogram (LBPH) algorithm for face recognition involves several steps to create an intermediate image that highlights facial characteristics and performs recognition based on histograms.

Applying the LBP Operation : The algorithm uses a sliding window approach with parameters like radius and neighbors to create an intermediate image that enhances facial characteristics. This involves comparing pixel intensities in local neighborhoods to create binary patterns.

Extracting Histograms: The intermediate image is divided into grids, and histograms are extracted from each grid. Each histogram represents the occurrence of pixel intensities in that region. These histograms are concatenated to create a final, comprehensive histogram representing the entire image.

Performing Face Recognition: During the recognition phase, the algorithm com-

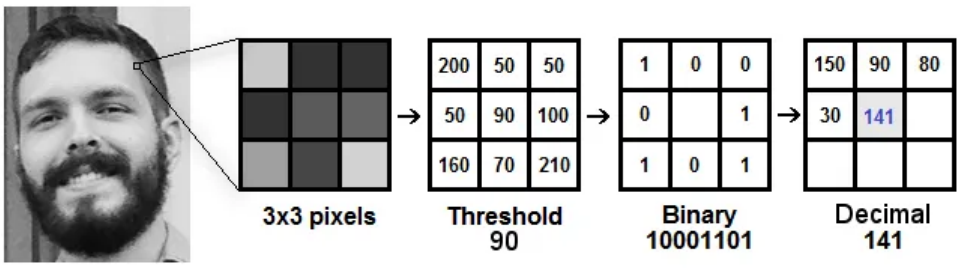

Figure 2.1: Picture pixel Histogram representation

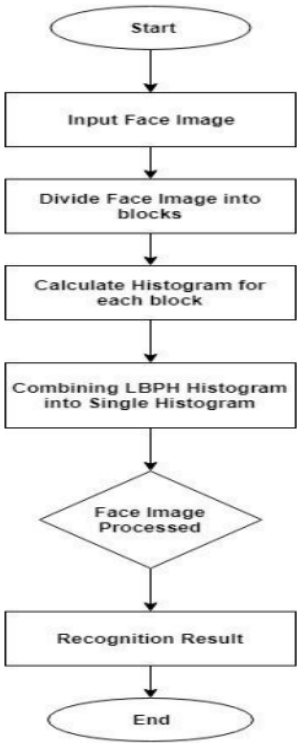

Figure 2.2: LBPH Algorithm Flowchart

pares the histogram of the input image (to be recognized) with histograms of images in the training dataset. This is done using a distance metric like Euclidean distance to find the closest match.

Thresholding and Confidence: The algorithm calculates a confidence score based on the distance between histograms. A lower confidence indicates a closer match. A threshold can be set to determine if the algorithm's recognition is successful based on the confidence score.

2.1.2 Open-CV

OpenCV stands for Open Source Computer Vision Library. It is an open source computer vision as well as software library for machine learning. OpenCV is a powerful computer vision library that provides tools for face detection and recognition. This project aims to leverage OpenCV's capabilities to create a face recognition system that runs efficiently on a Raspberry Pi.

2.2 SVM

SVMs are used for clustering data into two categories according to maximum boundary geometry. SVMs are supervised learning models associated with learning algorithms, and they are used to analyse data and recognise patterns. Generally, the results from SVM classification algorithms are more accurate than those derived from other ML approaches involving nonoptimised search methods, such as those involving artificial neural networks, least squares, k-nearest neighbour, Bayesian probability, and classification and regression trees, particularly when collect only limited training data. In an SVM training algorithm, examples are assigned to a category depending on whether nonlinear or linear binary classifiers are obtained from a set of training examples. SVMs have been shown to be useful tools for performing clustering and classification analyses [7]. In particular, SVM theory has been developed gradually from linear SVCs to hyperplane classifiers [8]; in other words, SVMs can efficiently perform nonlinear classification by using a kernel function, and their inputs can be mapped into high-dimensional feature spaces by selecting an appropriate kernel function. Furthermore, a favourable classification result is achieved by using a hyperplane that is the farthest from the nearest training data point of any class (Wikipedia 2015). A traditional linear SVM has a major drawback: assuming that the training data are linearly separable, the SVM cannot be valid when applied to practical real cases. Bernhard et al. (1992) suggested a novel approach to generating nonlinear classifiers; in the approach, a kernel function is used to obtain the maximum margins of hyperplanes. The nonlinear classification algorithm is formally similar to the linear SVM, except that each dot product is replaced by a kernel function.This enables the algorithm to map the maximum margin hyperplane in a transformed feature space.

2.2.1 Haar-Cascade

Haar Cascade classifiers are an effective object detection method proposed by Viola and Jones in 2001. It is a machine learning-based approach where a cascade function is trained from a lot of positive (faces) and negative (non-faces) images. It is then used to detect objects in other images. To use Haar Cascade classifiers for face detection, you need to train the classifier with positive and negative images, and then extract features from them. OpenCV provides tools for both training and using these classifiers.

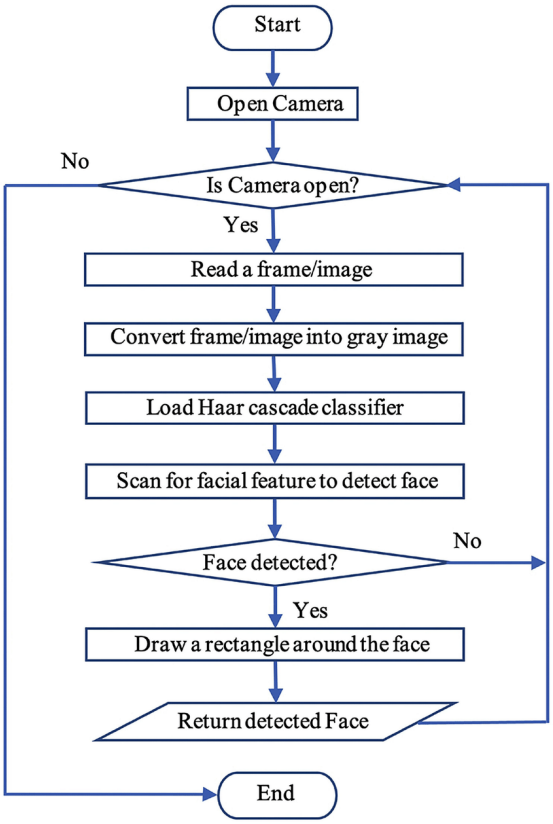

Figure 2.3: Workflow of Haar Cascade

Chapter 3
Methodology and Experimental Setup

3.1 Methodology

In this project, our software part was delineated into three pivotal phases, each essential for the successful realization of our objectives. We did our project using 2 methods and compared the results obtained from both. The following phases represent the structured backbone upon which our project was built:

1. Face Detection and Data Gathering
2. Train the Recognizer
3. Face Recognition

3.1.1 Face Detection and Data Gathering

In face recognition, we are implementing a face data collection tool using a webcam and OpenCV library in Python. The script captures images from the webcam, detects faces using a Haar Cascade classifier, and saves the detected faces into a dataset folder [3.1] for later use in training a face recognition model. We prompt the user to enter a numeric ID for the person whose face is being captured. This ID will be used to label the captured images. We continuously read frames from the webcam and convert them to grayscale for face detection. The detect MultiScale method is used to detect faces

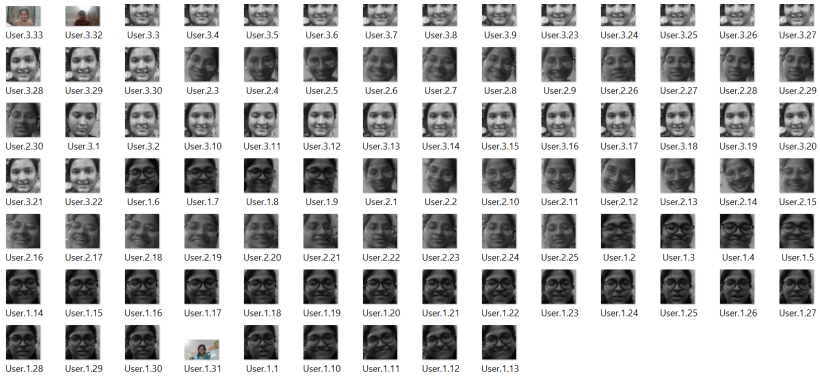

Figure 3.1: Dataset

in the grayscale frame. For each detected face, a rectangle is drawn around it, and the face is saved as an image in the dataset folder. The program exits when either the user presses the 'ESC' key or when 30 face samples have been captured.

3.1.2 Train the Recognizer

In 1st method, we are training the model using the LBPH (Local Binary Patterns Histograms) algorithm. The training process involves the following steps. For each detected face, we extract the region of interest (ROI) from the image and add it to the facesamples list. We extract the label (ID) from the image file name and add it to the ids list. Each face in the dataset is associated with a unique ID. We use the train method of the LBPH recognizer to train the model using the extracted face samples (faces) and their corresponding IDs. The trained model is saved to a file using the write method of the recognizer. Finally, we print the number of unique IDs (faces) trained and exit the program.

In 2nd method, we are implementing Face Recognition using Support Vector Machine. The function loads images from the dataset folder, converts them to grayscale, resizes them to a uniform shape (e.g., 100x100), flattens them into 1D arrays, and normalizes the pixel intensities to a range of [0, 1]. Further, we are splitiing into training and testing. We create an SVM classifier with a Radial Basis Function (RBF) kernel and use PCA for dimensionality reduction. The makepipeline function creates a pipeline that first applies PCA and then fits the SVM model.We fit the model using the training data and make predictions on the test data and calculate the accuracy.

3.1.3 Face Recognition

In the final phase of our project, we used the trained recognizer to make predictions on freshly captured faces. The script loads a pre-trained LBPH face recognizer from the 'trainer/trainer.yml' file. It initializes face detection.Inside the main loop, the script captures frames from the webcam, converts them to grayscale, and detects faces using the Haar Cascade classifier. For each detected face, it predicts the ID of the person and calculates the confidence level of the prediction. If the confidence level is below 100, it displays the name of the recognized person along with the confidence percentage. If the confidence level is 100 or above, it displays "unknown" as the name. The script continuously displays the video feed from the webcam with rectangles drawn around the detected faces and text showing the recognized person's name and confidence level. The function keeps track of the recognized names in an array and uses the Counter class

to count the occurrences of each name. It then determines the name that occurs most frequently and prints it to the console. The function exits the program and cleans up resources after capturing a specified number of frames (controlled by the count variable) or when the 'ESC' key is pressed.

3.1.4 Model Evaluation

The performance of the SVM-based image recognition system was evaluated using metrics such as accuracy, precision, recall, and F1-score. We also employed techniques such as confusion matrices to assess the classifier's performance across different classes.

3.2 Experimental Setup

3.2.1 Dataset Description

Our dataset comprises images belonging to our Team Members. A Total of 3 datasets are provided with each dataset consisting 30 pictures of one person. We divided the dataset into training and testing set to evaluate the performance of SVM classifier.

3.2.2 Parameter Tuning

We experimented with different kernel functions (e.g., linear, polynomial, radial basis function) and regularization parameters to determine the optimal configuration for the SVM models. We proceeded with Radial Basis Function in our model.

3.2.3 Hardware Implementation

Hardware Implementation[3.4] involves implementing our algorithm on Embedded System i.e. Raspberry Pi [3.2]. This involves few steps such as installing the latest Raspbian OS on our Raspberry Pi and ensure it is up to date. Connect the Pi to a monitor, keyboard, and mouse. Installing OpenCV on Raspberry Pi OS. Using the Pi camera module or a USB webcam[3.3] to capture images for training and recognition. Preprocess these images by converting them to grayscale and resizing them to a uniform size. Use the captured images to train the face recognition model. Using the LBPHFaceRecognizer class in OpenCV for this purpose. Display the recognized faces on the screen along with their names or IDs.

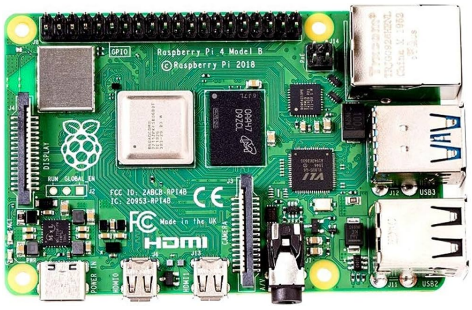

Figure 3.2: Raspberry Pi

Figure 3.3: Webcamera

Figure 3.4: Hardware Implementation

3.2.4 Results Obtained

The following results of accuracy were obtained by splitting the datasets into training and testing for both the algorithms:

```
main × data ×
 [INFO] 3 faces trained. Exiting Program

 [INFO] Accuracy: 95.65%
Confusion Matrix:
[[19  1  0]
 [ 1 11  0]
 [ 0  0 14]]
Precision: 0.96
Recall: 0.96
F1-score: 0.96
```

Figure 3.5: Results obtained from LBPH Algorithm

```
svm × data ×
Accuracy: 0.9782608695652174
Confusion Matrix:
[[16  0  1]
 [ 0 15  0]
 [ 0  0 14]]
Precision: 0.98
Recall: 0.98
F1-score: 0.98

 [INFO] 3 faces trained. Exiting Program
```

Figure 3.6: Results obtained from SVM algorithm

The results obtained from SVM algorithm are much better than LBPH algorithm.

Conclusion

In conclusion, we have explored two prominent face recognition algorithms, namely LBPH and SVM, and implemented them using OpenCV on a Raspberry Pi.

The LBPH algorithm proved to be a simple yet efficient solution for recognizing both front and side faces. It operates by labeling pixel intensities and creating a binary representation of the image, which is then combined with a histogram to characterize facial images.

On the other hand, SVMs offer a robust method for pattern recognition, particularly in distinguishing between faces and non-faces. Despite the challenges posed by the similarity of faces, SVMs excel in discriminating between different persons based on their facial features.

In our experimental setup, we collected datasets of team members' images and divided them into training and testing sets. By training our models on these datasets, we were able to achieve significant accuracy in face recognition.

The hardware implementation on Raspberry Pi showcased the feasibility of deploying face recognition systems on embedded systems. By leveraging the capabilities of OpenCV and Raspberry Pi, we demonstrated a practical approach to real-time face recognition.

In conclusion, the results obtained from the SVM algorithm outperformed those from the LBPH algorithm, showcasing the effectiveness of SVMs in face recognition tasks. This project highlights the potential of face recognition technology and its applications in various fields, from security to personalized user experiences.

References

[1] R. Ebrahimpour, N. Sadeghnejad, A. Amiri, and A. Moshtagh, "Low resolution face recognition using combination of diverse classifiers," in *2010 International Conference of Soft Computing and Pattern Recognition*. IEEE, 2010, pp. 265–268.

[2] H. Baqeel and S. Saeed, "Face detection authentication on smartphones: End users usability assessment experiences," in *2019 International Conference on Computer and Information Sciences (ICCIS)*. IEEE, 2019, pp. 1–6.

[3] P. Dinkova, P. Georgieva, A. Manolova, and M. Milanova, "Face recognition based on subject dependent hidden markov models," in *2016 IEEE international Black Sea conference on communications and networking (BlackSeaCom)*. IEEE, 2016, pp. 1–5.

[4] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Unsupervised learning of hierarchical representations with convolutional deep belief networks," *Communications of the ACM*, vol. 54, no. 10, pp. 95–103, 2011.

[5] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton, "On deep generative models with applications to recognition," in *CVPR 2011*. IEEE, 2011, pp. 2857–2864.

[6] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," 2010.

[7] W.-H. Lin, P. Wang, and C.-F. Tsai, "Face recognition using support vector model classifier for user authentication," *Electronic Commerce Research and Applications*, vol. 18, pp. 71–82, 2016.

[8] Y. Shi, F.-L. Chung, and S. Wang, "An improved ta-svm method without matrix inversion and its fast implementation for nonstationary datasets," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 9, pp. 2005–2018, 2014.