Real Time Face Recognition

Final Report

Ayman Siddiqui 202118019
Bhavya Chandrasala 202118028
Pritha Thakkar 202118035
Ojesh Vyas 202118036

December 2022

Dhirubhai Ambani Institute of Information and Communication Technology
MSc. Data Science

Submitted in fulfilment of domain project

Supervisor: Shruti Bhilare

Abstract

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. It is the most widely used field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs. Different types of computer vision include image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching.

This report explains all the techniques and algorithms which we have used to develop this system like the AdaBoost learning algorithm, Haar cascade classifier, openCV methods. The first approach towards developing a system that detects and recognizes face is face detection and creating a dataset of the face of 20 subjects from which the model learns and is used for recognition. Moving on to the next phase which is recognition of the faces from the real-time input video, for that LBPHrecognizer is used and the name of the respective person is displayed else unknown is displayed. By using these methods our system is capable of recognizing 70 percent of the faces correctly.

Keywords— OpenCV, AdaBoost, Haar-cascade classifier, LBPHrecognizer

Contents

1 Intoduction

The use of video analytics technology has increased recently. It automates gathering various data based on the sequence of frames received from video cameras in real-time or from videos using computer vision algorithms. Video surveillance, security systems, trade, and transportation can all benefit from this technology. Detection is a computer technology related to computer vision and image processing that detects instances of semantic objects of a specific class (such as humans, buildings, or cars) in digital images and videos. For the Detection process in computer vision, there are various methods, and each one has a different level of accuracy according to their advancement level. Some methods were invented at a very early stage giving more cases of false detection compared to the advanced methods discovered after that. Face recognition from video streams can be done and directly applied to a system that uses face recognition, like security systems, identification systems, etc. To make such a model recognize the face, we can use multiple algorithms. Here in this report, we will understand the system using the Ada-boost learning algorithm and Haar cascade classifier and predict using LBPHFace-Recognizer. Moreover, next is task analysis and the creation of requirements for the system. These criteria allow us to outline the application's working algorithm and explicitly identify the necessary functionality. This information includes the selection of development tools, a detailed description of the program implementation of the person recognition system, and testing of the system's performance with various settings for recognition methods and sets of indicators.

2 Problem Statement

A system that is able to detect the person's face and recognizes it by displaying the respective name.

3 Methodology

3.1 Dataset

To prepare the dataset for a system that is able to detect and recognize the face we have to capture and store the face image of the respective person after preprocessing, which is then used for training the model. The training data folder contains one folder per person, and each folder has the person's name. The images are stored after converting RGB into Grayscale and resizing it from the original dimension of (1440,960) to the resized dimensions (112, 92) with labels. Typically, many images are used to train facial recognition to learn the different faces of the same person. For example, with glasses, without glasses, laughing, sad, happy, crying, with a beard, without a beard, etc. The more images you use to train, the better. Keeping this in mind, we have taken 90 images of a person in two sessions. So our training data consists of about 20 people with 90 images of each person. The preparation of the data phase was divided into the following sub-steps. Read all the person's labels in the training data folder, which was captured. Each item has a label number, where the label represents an integer representing the label assigned to this subject.

3.2 Tools and Techniques

To detect the face of a human from real-time video, we can use the following techniques and algorithms.

3.2.1 OpenCV

A well-known computer vision library, OpenCV (Open Source Computer Vision), was created by Intel in 1999. Real-time image processing is the main focus of the cross-platform library, which also has patent-free versions of the most recent computer vision algorithms. Since Willow Garage has been providing support since 2008, OpenCV 2.3.1 has an Android, C, C++, and Python programming interface. Because OpenCV is distributed under a BSD license, it can be used for commercial and academic projects. The brand-new FaceRecognizer class for face recognition is now included in OpenCV 2.4. recognition.

3.2.2 Ada-Boost Learning Algorithm

AdaBoost is an ensemble method in machine learning, which is an iterative algorithm. The fundamental concept is to combine the weak classifiers into a more robust classifier using the same training set to generate various classifiers (weak classifiers like Decision stump) and (strong classifiers like SVM, Decision Tree). This algorithm is implemented by altering the distribution of the data, and each sample's weight is verified based on whether or not the training set's samples were correctly classified and whether or not the previous overall classification was accurate. The following classifier will train on the new dataset. All of the classifiers will ultimately be combined for a final classification.

Instead of reducing the image approach, the AdaBoost algorithm's face detection mechanism enlarges the detection window approach of multi-scale detection mechanisms. Because each time the image is reduced, you must calculate the Haar features, which requires a significant amount of calculation and takes a long time. The AdaBoost algorithm's real-time processing requirements are challenging to meet. The detection window starts with the same size as the sample

size, but it moves to cover the entire image area following a specific scale parameter and marks the face area. After the entire image has been iterated, the detection window will be expanded in terms of the specified amplification parameters for the following iteration. The processing will be executed repetitively until the size Of the detection, of the window, is more than half the size of the image.

3.2.3 Haar-Cascade Classifier

It is an object detection algorithm used to find faces in still photos or moving videos. The algorithm uses Viola and Jones's suggested edge or line detection features. These image characteristics make it simple to identify the image's edges or lines and regions where there is a sharp change in the pixel intensities.

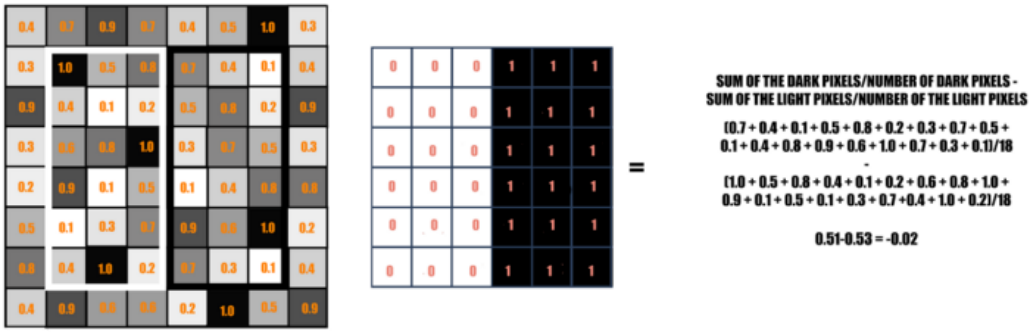

Figure 1: The rectangle on the left is a sample representation of an image with pixel values 0.0 to 1.0. The rectangle at the center is a haar kernel which has all the light pixels on the left and all the dark pixels on the right. The haar calculation is done by finding out the difference of the average of the pixel values at the darker region and the average of the pixel values at the lighter region. If the difference is close to 1, then there is an edge detected by the haar feature.

The goal is to calculate the sum of all the image pixels in the haar feature's darker and lighter areas. Then ascertain what makes them different. The haar value will be closer to 1 if there is an edge in the image separating light pixels on the left from dark pixels on the right. In other words, if the haar value is closer to 1, we say that an edge has been detected since the haar value is far from 1.

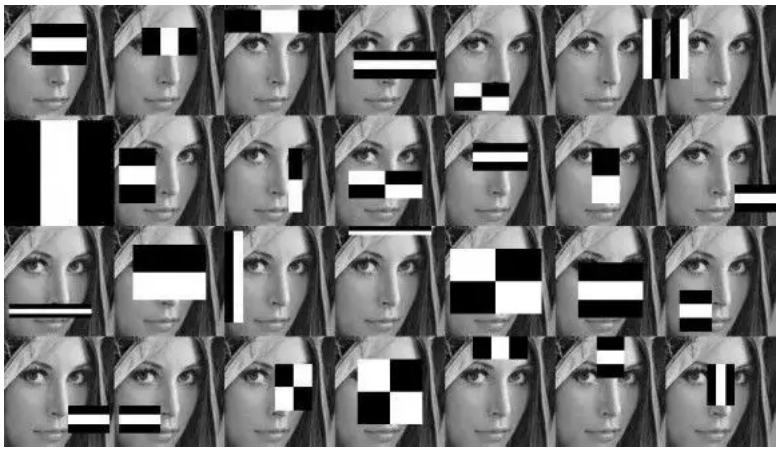

Figure 2: Haar features

In order to find the specific feature, the haar feature continuously moves from the top left of the image to the bottom right as shown above in figure 4. This is merely a visual representation of the haar feature traversal as a whole. The haar feature would actually move pixel-by-pixel across the image when doing its actual work. Additionally, the haar features will be applied in all possible sizes.

3.2.4 LBPH Face Recogniser

A visual descriptor style called Local Binary Patterns (LBP) categorizes computer vision. LBP first had a representative in 1994. Since then, it has been discovered to be a reliable component for categorizing texture. In more detail, once LBP and the descriptor histogram of oriented gradients are combined (HOG). It improves the execution of identification on

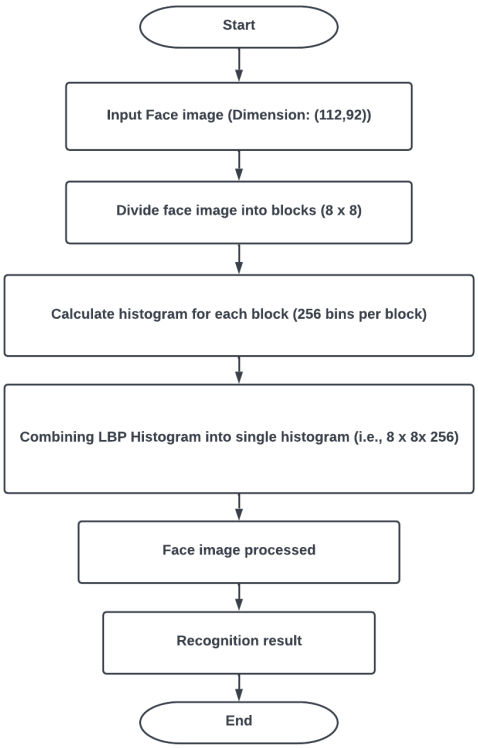

Figure 3: LBPH Recogniser Flow

some datasets. The flowchart diagram for the LBPH algorithm is shown.

Only 256 positions (0–255) representing the occurrences of each pixel intensity will be present in each histogram (from each grid). To make a new, larger histogram, we concatenate each histogram. The characteristics of the original image are reflected in the final histogram. To encode features, the image is divided into cells (8 x 8 pixels). By using a clockwise or counterclockwise bearing of the surrounding pixel values, it is contrasted. Each neighbor's intensity value is compared to the value of the main pixel. Depending on whether the difference is higher or lower than 0, or both, the location is given either a 1 or a 0. A single cell receives an 8-bit value as a result. The calculation of the matrix's middle element in comparison to its surrounding elements. In order to create a histogram that accurately represents an image, we repeat the process for a new image after receiving an input image. Therefore, all required to locate the image corresponding to the input image is to compare two histograms and return the image with the closest histogram. The distance between two histograms can be calculated using various methods, such as the Euclidean distance, chi-square, absolute value, etc.

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

3.3 Training and Testing

3.3.1 Detector

```
detector = (cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
Cv2.VideoCapture()
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
mini = cv2.resize(gray,(gray.shape[1] // size, gray.shape[0]//size))
faces = detector.detectMultiScale(gray, 1.3, 5)
cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

We used the Haar-like features classifier cascade to identify a face, which is part of the OpenCV library. A parameter file is required in XML format for the correct operation of the classifier. Creating this "knowledge" file is a rather complicated task. So, for the parametrization of the method, we used the ready classifier haar cascade frontal face default.xml available from the link.

The detected faces from the real-time video are then captured and saved in a destination folder after resizing the

dimensions from (1440,960) to (112,92) and converting them into a grayscale image. Before using the face detector, we need to translate the image into a grayscale format.

The above method (last line of code) will return the linear size of the face image - x, y and height - h, width - w for all faces present in the image, and now you can select all faces and outline them with rectangles (x, y, w, h): for (x, y, w, h) in faces.

3.3.2 Detection

```
model = cv2.face.LBPHFaceRecognizer-create()
model.train(images, lables)
face-cascade = cv2.CascadeClassifier(haar-file)
prediction = model.predict(face-resize)
cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
if prediction[1] <200:
cv2.putText(im, '%s - %.0f' % (names[prediction[0]], prediction[1]), (x - 10, y - 10),
cv2.FONT-HERSHEY-PLAIN, 1, (0, 255, 0))
else:
cv2.putText(im, 'Unknown', (x - 10, y - 10), \\cv2.FONT-HERSHEY-PLAIN, 1, (0, 255, 0))
```

You must use a preset dataset to train the algorithm to detect faces to do face recognition. In the preceding example, we built a dataset for our face recognition system; now, we need to use this dataset to train facial recognition using OpenCV Python. Next is to initialize the recognizer and face detector.

The algorithm mentioned above uses the predict () method to identify a face. The name of that specific in- dividual (Eg. Bhavya) is displayed if the prediction rate is below 200, and "Unknown" will be displayed next to the rectangle if it is above 200. We are compiling a collection of photos and the names that go with them using the im- ages, labels, titles, and IDs. To retrieve a list of all the files and folders in a given directory, use a program's listdir() method.

The LBPH Algorithm is another component of this application. The LBPH (Local Binary Patterns Histograms) algorithm is used here to find faces. By thresholding the area around each pixel and treating the result as a binary number, it labels the pixels in a picture. We use the Waitkey() method after facial recognition to save the delay time for that window. Key variables contain the value returned by the waitKey() method. The loop is broken, and the output frame is destroyed if the key value is "q."

4 Experiments and Result

After creating the dataset and training the model based on the dataset, we can see it correctly identifies the persons and displays the name of the respective detected person. As you can see, it generates outputs with the name (Bhavya, Ayman, Pritha, Bhumi)

Now to check the system's accuracy, we have tried to recognize the faces in a group. Under different conditions, we have achieved 70 percent accuracy, as our system can recognize 6-7 people in a group. In comparison, when the accuracy of an individual is getting detected is 90 percent. Moreover, we can also increase the performance by training the model by creating the dataset with a higher number of images. We can also use different classifiers and recognizers to improve the system's efficiency.

5 Conclusion

There are numerous areas where recognition performance can be enhanced, some of which are relatively simple to execute. Some examples are the processing of colors, eye detection, upper- and lower-body detection, etc. By clicking more photos of each individual, mainly from various perspectives and lighting situations, you can increase facial recognition accuracy by using more input images. We may infer from the data that this program performs best when the subject is close to the camera. This can also improve human-machine interaction so that computers can read people's faces and be used for attendance marking etc.

References

[1] V. Gupta, D. Sharma, "A Study of Various Face Detection Methods," International Journal of Advanced Research in Computer and Communication Engineering, vol. 3(5), pp. 6694–6697, May 2014.

[2] P. Viola, M. Jones, “Rapid object detection using a boosted cascade of simple features”, IEEE Computer Society CVPRW, vol. 1, pp. 511 – 518, December 2001

[3] Y. Freund, R. E. Schapire, “A Short Introduction to Boosting,” Journal of Japanese Society for Artificial Intelligence, vol. 14(5), pp. 771 – 780, 1999 Face detection

[4] https://www.analyticsvidhya.com/blog/2 019/03/OpenCV- functions-computer-vision- python/

[5] HTTPS:// www. geeksforgeeks.org/ python- programming- language

[6] https://www.academia.edu/Documents/in/Image Processing with OpenCV

[7] https://www.techopedia.com/definition /32071/facial recognition

[8] https://stackoverflow.com/questions/5 3926657/215 assertion failed src empty in function cvcvtcolor