Design Document

For

Face Authentication System

Prepared by: Palash Bajpai
Roll No.: B180759CS
Guided by: Dr. Lijiya A
Course: CS4046D Computer Vision
Date: 30 November 2021

Table of contents

Contents

1. Abstract

Face recognition is one of the most advanced biometric technologies available. Cameras are being used with an increasing frequency to identify people and track their whereabouts. Already in use in some major airports and public spaces, these systems use CCTV cameras to collect data on human faces. This process includes the shape of the face, Location of facial features, and other characteristics that make a person’s face unique. Some face recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. It’s a process that has come a long way over time. In early generations of facial recognition, systems were limited to identifying one person from a lineup. Nowadays, these systems can identify people from big crowds and even in night-time images.

2. Problem Statement

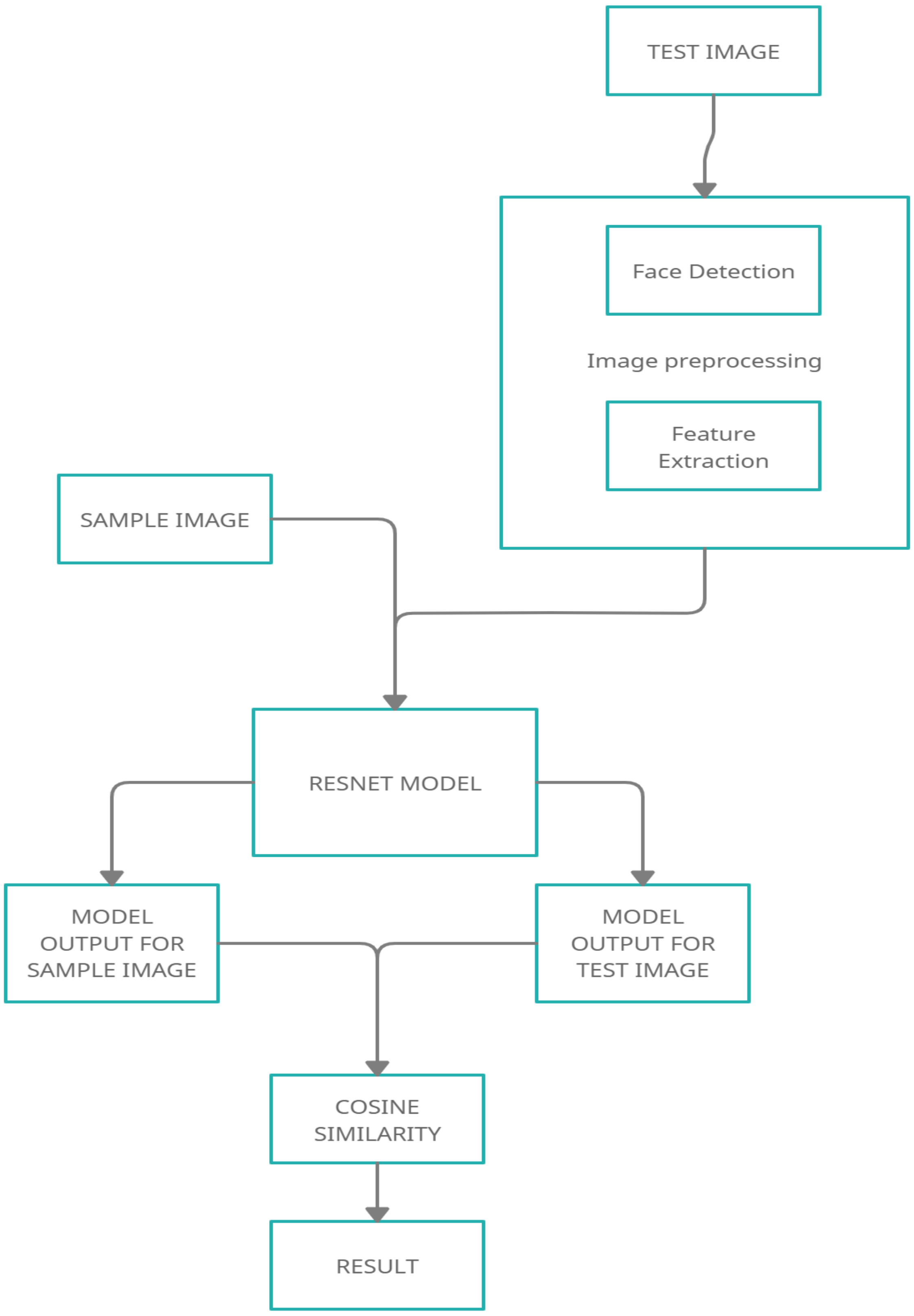
To build a Classifier that can be used for face Authentication. Input: An image containing the front face photo of a person. Output: ”Yes” if the image is containing your face and ”No” otherwise.

3. Model Design

3.1. Technology and Libraries Used

1. Python as our programming language.
2. Opencv to capture and preprocess images.
3. Keras for our deep learning model.
4. Sklearn to perform cosine similarity.

3.2 Design Diagram



3.3 Model Architecture

1. **Face Detection and Feature extraction**

   Since we only need to compare between the faces and not the body, we will first crop out the face of the person from the image and then only we will compare it with the sample image. To detect face we have used haar cascade face detection. To include this model we have used ' haarcascade frontalface default.xml' file which contains a pretrained model for face detection. Haar Cascade classifier is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses “integral image” concepts to compute the “features” detected. Haar Cascades uses the Ada-boost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers then uses cascading techniques to detect the face in an image. Haar cascade classifier is based on the Viola-Jones detection algorithm which is trained in giving some input faces and non-faces and training a classifier that identifies a face.

   AdaBoost is used to get the best features among all these possible features. These features are also called as weak classifiers. After these features are found a weighted combination of all these features is in use in evaluating and deciding any given window has a face or not. Each of the selected features (weak classifiers) is considered okay to be included if they can at least perform better than random guessing (detects more than half the cases). Each of the weak classifiers is relevant in detecting a part of the face. The output of a weak classifier is binary if it has identified a part of the face or not. Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

2. RESNET MODEL

We have used a pre-trained ResNet model for finding features of our image. ResNet-50 is a convolutional neural network that is 50 layers deep. We have loaded a pre-trained version of the network trained on more than a million images from the ImageNet database. We have not included the last softmax layer of the Resnet model, hence this modified model does not classify the image into predefined classes, it just returns an array containing weights of different features.

We used Resnet since it gave the best accuracy from all defined models.

3. COSINE SIMILARITY

We have used cosine similarity to compare the model outputs from the sample image and test image. Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.

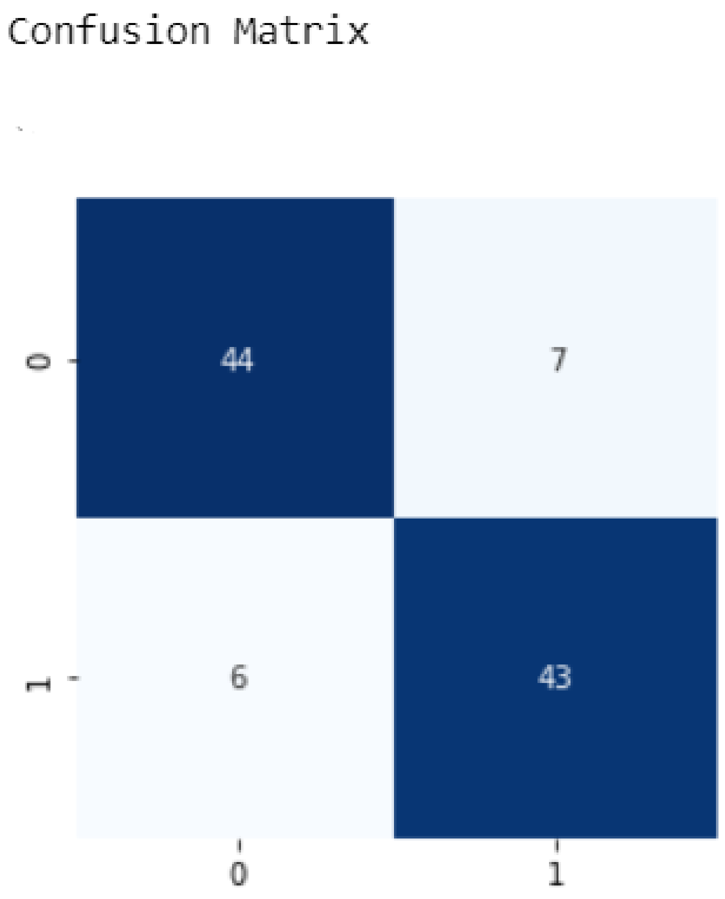
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Cosine Similarity is a value that is bound by a constrained range of 0 and 1. The similarity measurement is a measure of the cosine of the angle between the two non-zero vectors A and B.Suppose the angle between the two vectors was 90 degrees. In that case, the cosine similarity will have a value of 0; this means that the two vectors are orthogonal or perpendicular to each other. As the cosine similarity measurement gets closer to 1, then the angle between the two vectors A and B is smaller. The images below depict this more clearly.

3.4 Model Comparision

We found that the best threshold was: 0.65

With our RESNET50 model, we got accuracy : 0.87



CONFUSION MATRIX (RESNET50)

Accuracy with different models :

RESNET50	:	0.87
VGG16	:	0.7
VGG19	:	0.75