Face recognition & face mask detection

Written and coded by:

Tomer Maabari

Ortal Hanoch

Mulugeta Fanta

Software Design Document (SDD)

Table of Contents

1. Introduction

1.1 Purpose

The project purpose is to create a system to identify persons and objects via face recognition.
This design document presents the designs used or intended to be used in implementing the project.
The designs described, follow the requirements specified in the Software Requirements Specifications document prepared for the project.
This is an abstract SDD document that many of these are still not defined and might be change over time.

1.2 Scope

Face recognition, the software allows this with the help of a live feed webcam.

Availability, the advantage of the software is that it is accessible to the customer and can be used from anywhere.

Easy to use, the software is easy to use and provides instant results in a way that is easy to see.

1.3 Overview

The face recognition system will be able to analyze variety of different faces including faces with many accessories and different features (skin color, glasses etc.).
The user will be able to create their own image database.

1.4 References

https://www.cs.fsu.edu/~lacher/courses/COP3331/sdd.html
Cascade Data From: (XML format)
https://github.com/opencv/opencv/tree/master/data/haarcascades

1.5 Definitions and Acronyms

- TensorFlow
- CNN – Convolutional Neural Network
- Haar-Cascade Detection Algorithm
- OpenCV library
- Matplotlib

2. System Overview

TensorFlow - Open-source library for machine learning, building, and training neural networks.

CNN - Class of deep neural networks most commonly to analyzing visual imagery, CNNs are regularized versions of multilayer perceptron. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer.

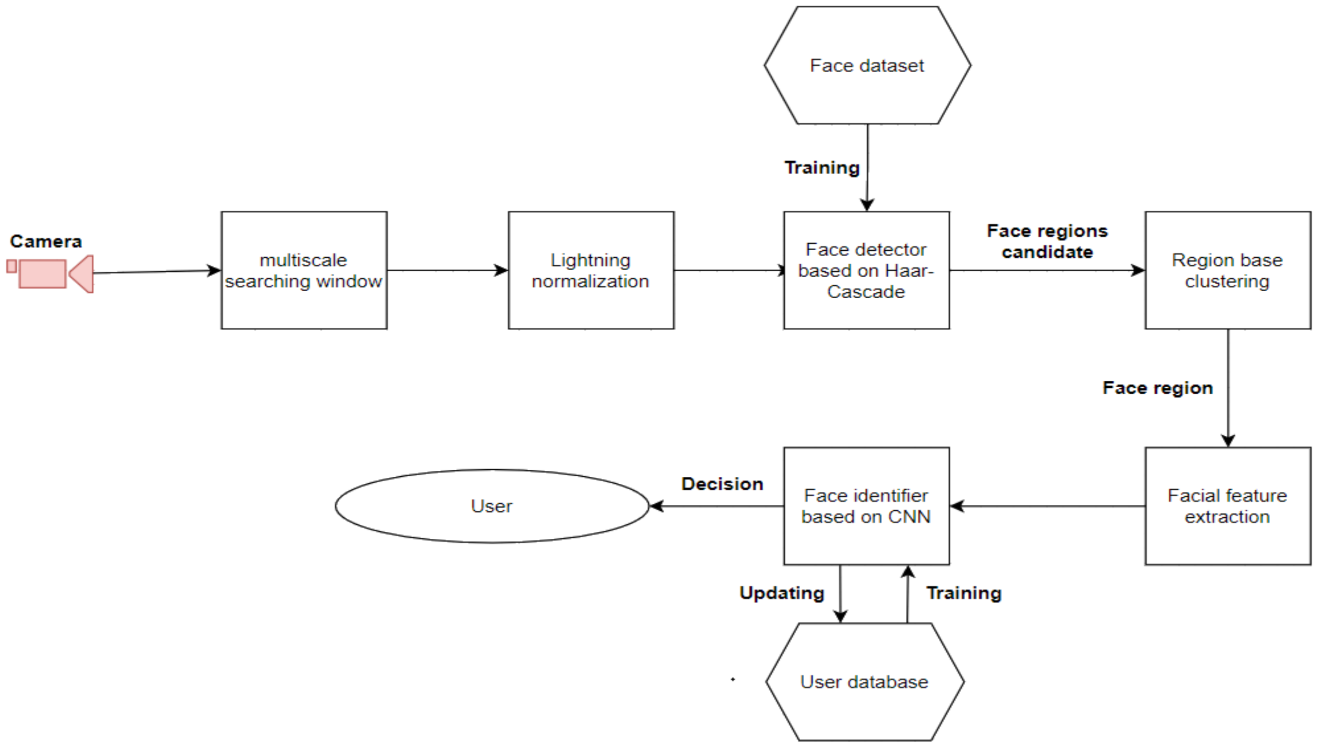
Object Detection using Haar feature based cascade-classifiers is an effective object detection method, here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier.

OpenCV- a software package designed to help develop computer vision applications, the library is primarily targeted for real-time computer vision applications.

Matplotlib- a comprehensive library for creating static, animated, and interactive visualizations in Python.

3. System Architecture

3.1 Architectural design



3.2 Decomposition Description

Our design includes three main processes:

- OpenCV- image processing using webcam.
- Haar Cascade Algorithm- a function that trained from a lot of positive and negative images. It is used to detect objects in other images.
- DB storage will be chosen in the future.

3.3 Design rational

The most important reason we chose the diagram on section 3.1 is because it is easy to follow the flow chart to understand our idea and intention. The critical issues are to learn from the data and update the model for a better result (CNN).

4. <u>Data Design</u>

4.1 **Data Description**

We are having an image databases folder that are divided to each individual feature. The information domain of our system is transformed into data structures using numpy array. The major data are images that are converted to numpy array.

4.2 **Data Dictionary**

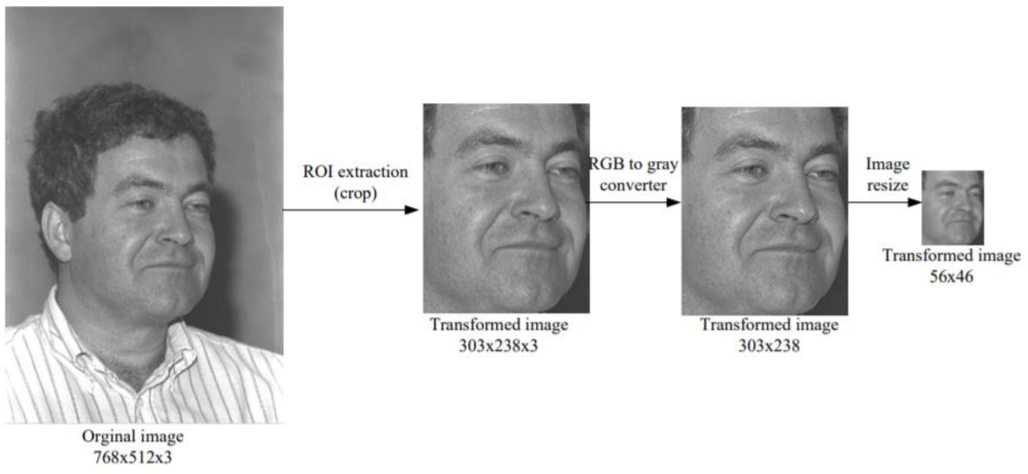
Our Dataset contains thousands of face images. We will need to use them and storage then in efficient way.

- Separate files- each image will belong to a folder by specific attributes (Until this day we separate images only by mask).
- User's Database.

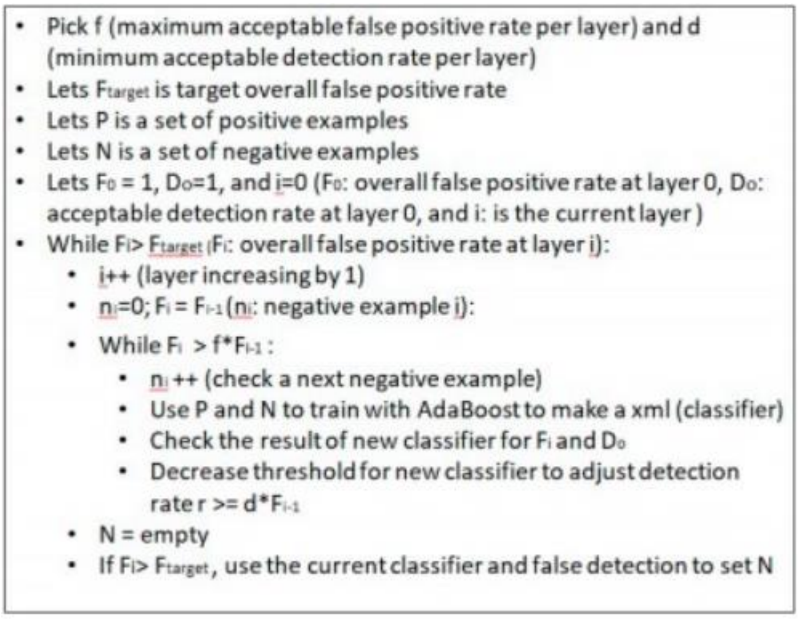
5. <u>Component Design</u>

In this section we will explain in detail what each component does in a more systematic way.

- OpenCV – Using this tool has given us the ability to handle and process any image in any quality, preforms image reading from the folder or camera and manipulates the image so that we can use the neural network (Change the color, size), after that convert the image into a pixel matrix which can classify the image.

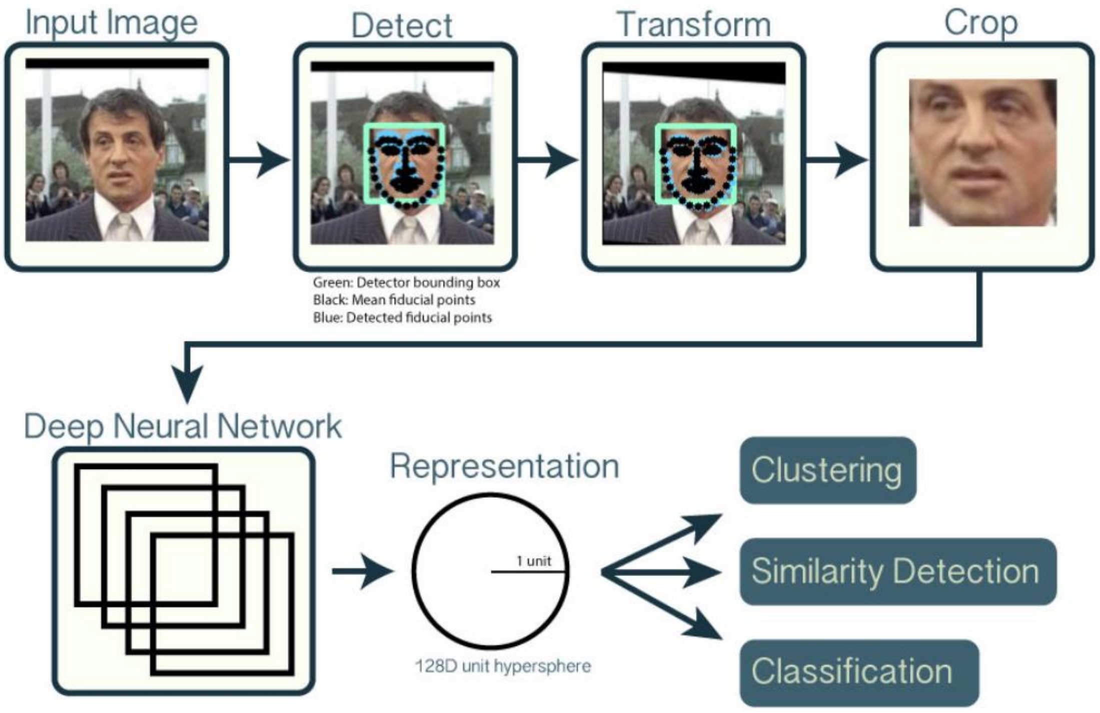


- Haar Cascade Algorithm – Built in library in OpenCV. This algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. The we need to extract features from it. For this, Haar features are shown as a single value obtained by subtracting sum of pixels.
  Here we represent example of pseudocode and diagram to illustrate the idea.



```
• Pick f (maximum acceptable false positive rate per layer) and d
  (minimum acceptable detection rate per layer)
• Lets $F_{target}$ is target overall false positive rate
• Lets P is a set of positive examples
• Lets N is a set of negative examples
• Lets $F_0 = 1$, $D_0=1$, and $i=0$ ($F_0$: overall false positive rate at layer 0, $D_0$:
  acceptable detection rate at layer 0, and i: is the current layer )
• While $F_i > F_{target}$ ($F_i$: overall false positive rate at layer i):
    • i++ (layer increasing by 1)
    • $n_i=0$; $F_i = F_{i-1}$ ($n_i$: negative example i):
    • While $F_i > f*F_{i-1}$ :
        • $n_i$ ++ (check a next negative example)
        • Use P and N to train with AdaBoost to make a xml (classifier)
        • Check the result of new classifier for $F_i$ and $D_0$
        • Decrease threshold for new classifier to adjust detection
          rate $r >= d*F_{i-1}$
    • N = empty
    • If $F_i > F_{target}$, use the current classifier and false detection to set N
```

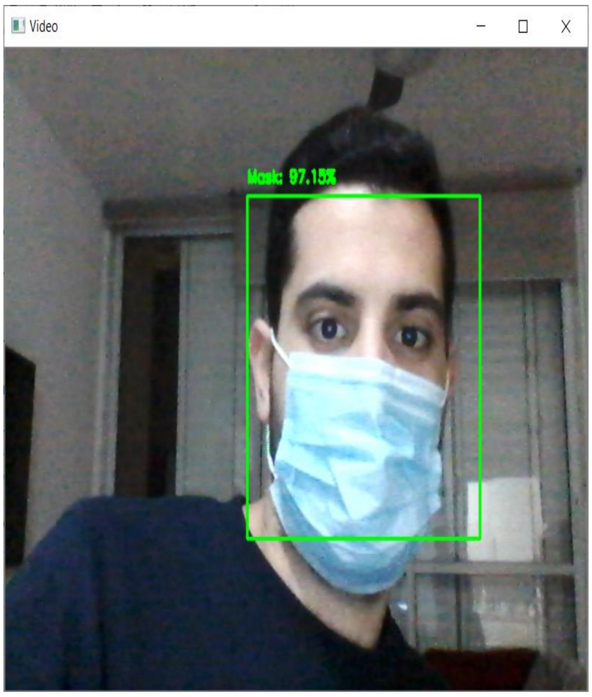
Figure 1. Haar-cascade algorithm pseudo code.

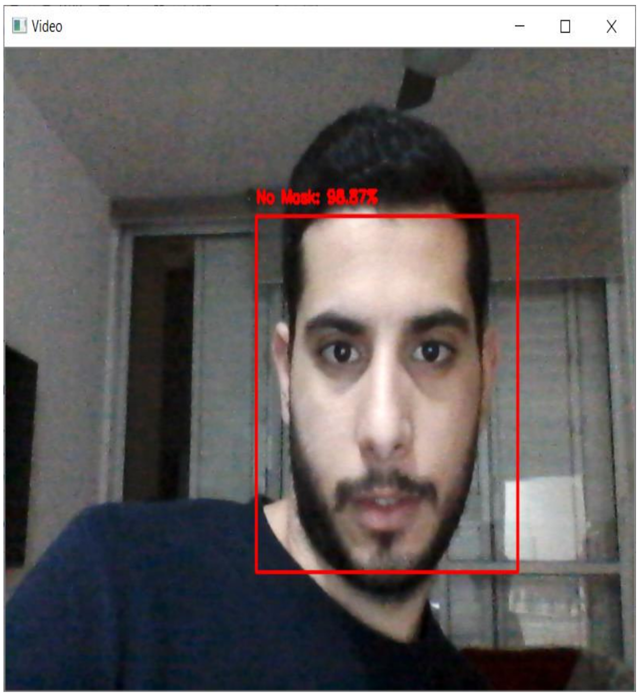


6. Human Interface Design

6.1 Screen Images

These images we provide are for the purpose of illustrating our product.


Video
Mask: 97.15%


Video
No Mask: 98.87%