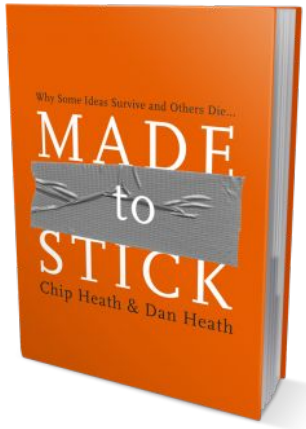
Eye and occlusion detection for liveness detection

A presentation by Parth Kalkar & Bekhruz Nutfilloev

Introduction

Our project focuses on automatically identifying eyes and occlusion for liveness detection.

In face detection and landmark detection applications, computer vision plays a significant role.

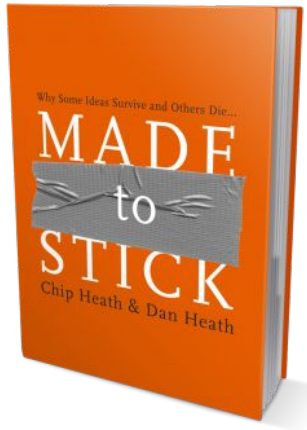


Methodology - CNN

CNN technology is used for eye detection.

To do this, we first need to use face detection technology . We use the MTCNN(onnx) face detection system.

Next, we will need to define the eye area. For this, dlib face landmark was used. The next step is to turn the right eye into the left eye by flicking it. Then the performance of the model will increase, we can work not with two eyes, but with exactly the same eyes.



MTCNN onnx: https://github.com/yiyuezhuo/mtcnn-onnxruntime

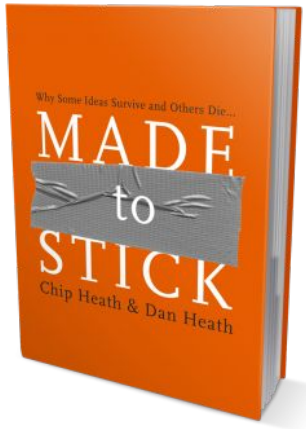
Methodology - YOLOv5

For occlusion, we use only YOLOv5.

First of all, pictures with blocked faces and regular faces are collected.

In the next step, two eyes, a nose, and a mouth are defined. Then, the symbol in the following sequence is selected. [eye, eye, nose, mouth](0, 0, 1, 2)

These characters are checked in all pictures. If any sign is not found, this image is defined as not passing the filter in occlusion detection.



Dataset creation

First, we need pictures of people with their eyes open and eyes closed. We compiled it ourselves (the GitHub repo hosts all datasets).

Furthermore, we divided them into separate folders for two classes. The next step is to cut out the eyes in each picture. First, it is necessary to identify the face from the images and identify the eye in the identified face. For this, we used the Dlib face 68 landmark model.

The points between 36:42 correspond to the left eye, and the points between 42:48 correspond to the right eye. These points separated the eyes, and we predicted them through the CNN model.

Dataset Labelling

For occlusion, we need random face images and images of certain objects with their eyes and mouth occluded.

We also collected these manually, all these datasets are listed in the GitHub repository. We labeled them using makesense.ai and LabelImg built-in programs.

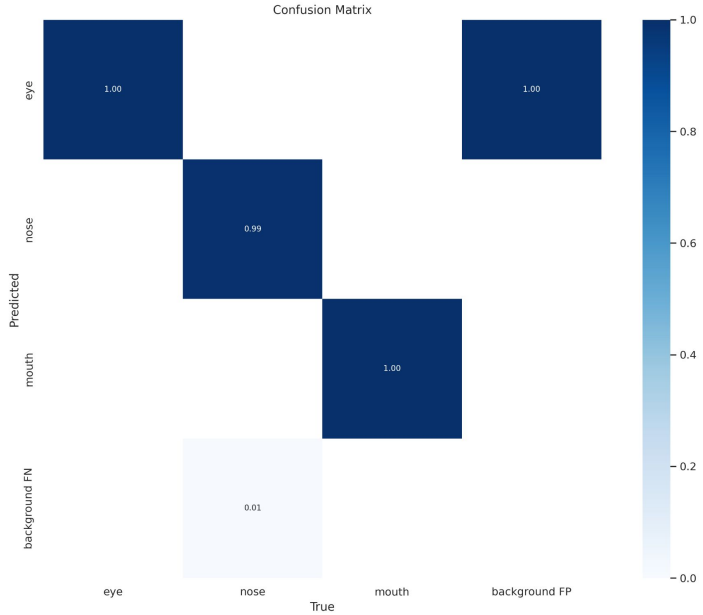
These are names: `['eye', 'nose', 'mouth']` by three classes. Furthermore, the train part was executed based on the Yolov5 architecture.

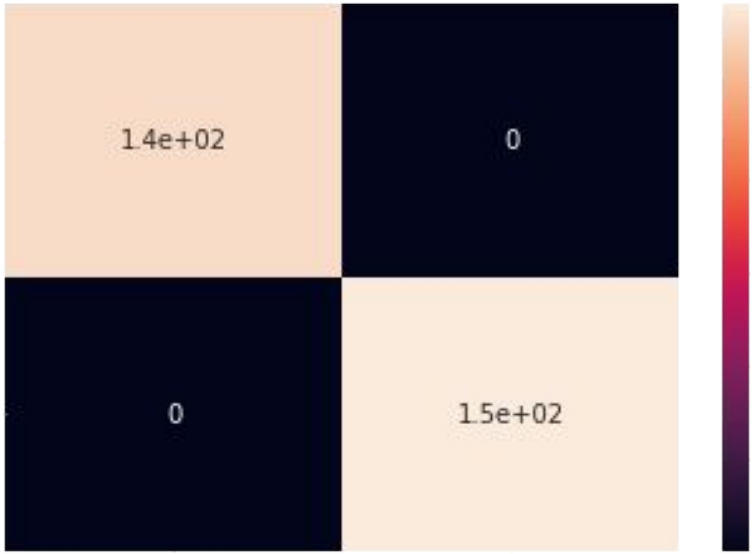
Training

- The training part of eye detection includes the following steps. First, the detected eye image is converted to grayscale and reshaped to 26x34x1 size, and 2 values are output. In this case we have 880,129 model parameters.

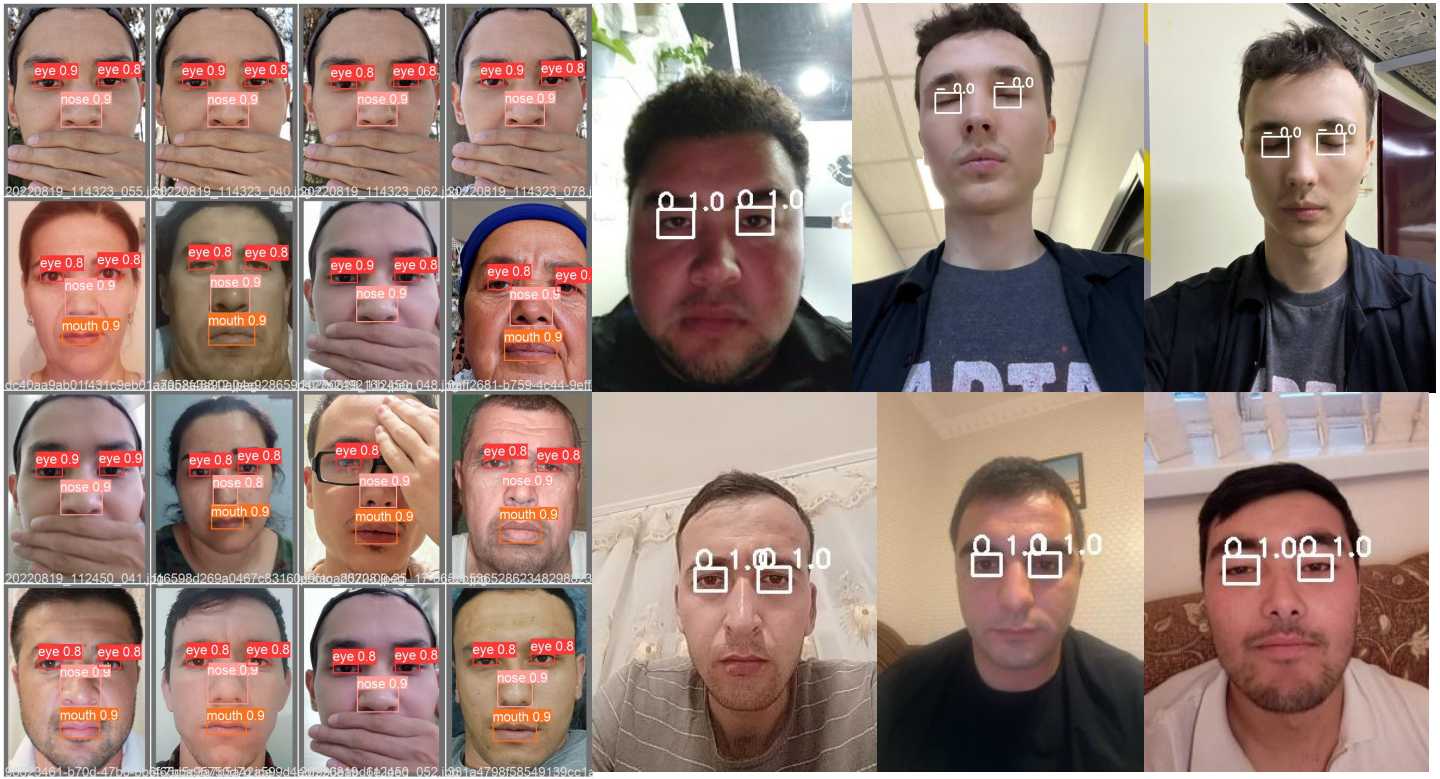
- The occlusion part accepts 640x640x3 photos. In this case, we used the YOLOv5s pretrain model. This is a small model and we found it to be the model that gave us the best accuracy and performance for our images. You will be able to see all the results and the code on our GitHub page.

Results


Confusion Matrix
eye
nose
mouth
background FN
Predicted
1.00
1.00
0.99
1.00
0.01
1.0
0.8
0.6
0.4
0.2
0.0
eye
nose
mouth
background FP
True


1.4e+02
0
0
1.5e+02

Results


eye 0.9 eye 0.8 nose 0.9
eye 0.8 eye 0.8 nose 0.9 mouth 0.9
eye 0.9 eye 0.9 nose 0.9
eye 0.8 eye 0.8 nose 0.8 mouth 0.9
eye 0.8 nose 0.9 mouth 0.9
eye 0.8 eye 0.8 nose 0.9 mouth 0.9
1.0 1.0
0.0 0.0

Thank you for your **Time**
and **Attention!**

GITHUB