CS 763: Group Project

Group: Blue_Onion

13 April 2023

1 Overview

As a final group contribution towards CS 763, we had to create a CV-based project that uses deep learning. We created a ***virtual proctoring system*** for this. Tired of being called the *covid batch*, the current sophomores and juniors have gone through a lot of virtual proctoring applications, ranging from Codetantra to even Zoom!
This project is a humble attempt to enforce the most important features of a virtual proctoring system.

2 Gaze tracking

Firstly, we started with OpenCV's facial recognition model. But it turned out to be very slow for practical reasons. Here, dlib came to our help with its 68-point face landmarks detector. The mask and bitwise-AND was the only logical step that we could come up with.

3 Multi-face detection

A lot of models were tried for this, ranging from the HoG detector to face_recognition package, but all of them failed when the person in the frame turned their head sideways. This was again taken care of by dlib. One thing is for sure, over this course, we saw a lot of beneficial features of dlib.

4 Mouth-opening detection

When we first thought of implementing this feature, we thought of a lot of approaches, ranging from YOLO to SSD. But then we realised that this is not just a simple object detection problem. We need to localise the mouth feature. Then we remembered dlib! So, we simply implemented the 68-point face landmark. The logic was easy to come up with, and so the overall implementation went smooth.

5 Cell phone detection

As the name suggests, this is a simple object detection task. Although we knew that YOLO can be used for this, we were experiencing a lot of problems with installation from the repository directly, to begin with. But then, we got to know that *pip* can be used to install YOLOv5. That was a great sigh of relief for us. After that, the object detection was an easy task.

6 Web app integration

We had thought that this would be the easiest task to accomplish. But we had a large obstacle waiting for us. The image matrix returned by OpenCV can not be directly rendered by Django/HTML. For this reason, after a lot of tinkering, we encoded the matrix to a JPEG, did a tobytes() on the same, and then we finally rendered the frame on to the webpage.
