A

SYNOPSIS

On the Topic

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING:

‘COMPUTER VISION’

(Submitted as a task for Synapse Interviews)

By- DISHANT ZAVERI

CONTENT OF SYNOPSIS

1.INTRODUCTION

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision is a broad term for the work done with deep neural networks to develop human-like vision capabilities for applications, most often run-on NVIDIA GPUs. It can include specific training of neural nets for segmentation, classification and detection using images and videos for data.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will "look" at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model "look" by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is "seeing." The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Computer vision can handle many more tasks. Developed with convolutional neural networks, computer vision can perform segmentation, classification and detection for a myriad of applications.

Computer vision has infinite applications. With industry changes from computer vision spanning sports, automotive, agriculture, retail, banking, construction, insurance and beyond, much is at stake.

2.DROWSINESS DETECTION

We are going to extend this method and use it to determine *how long* a given person's eyes have been closed for. If their eyes have been closed for a certain amount of time, we'll assume that they are starting to doze off and play an alarm to wake them up and grab their attention.
Fatigue and microsleep at the wheel are often the cause of serious accidents. However, the initial signs of fatigue can be detected before a critical situation arises
With this Python project, we will be making a drowsiness detection system. A countless number of people drive on the highway day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy.
The majority of accidents happen due to the drowsiness of the driver.
Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving.
The objective of this intermediate Python project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected.

3.PROBLEM STATEMENT

COMPUTER VISION

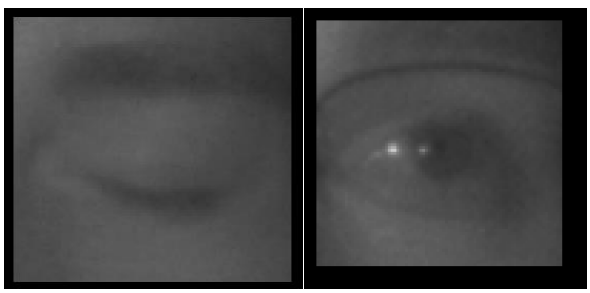
Given a dataset of human face images taken from a dashboard of a car, detect if the driver is drowsy or not. Note that a driver can be classified as drowsy if their eyes are closed more often than open.

Describe the pre-processing steps required on the image (if any).

Describe techniques you would use to detect the eyes in the entire image.

Describe techniques you would use to detect drowsiness in the eyes.

End the synopsis with a conclusion which should contain information about why you selected a particular algorithm over other existing solutions. Consider this as a supervised learning problem where you have been given images of open and closed eyes for training.

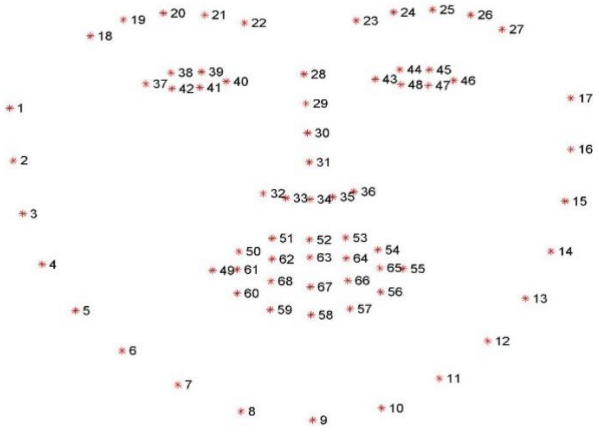


4.RESEARCH

Q1-Describe techniques you would use to detect the eyes in the entire image.

Ans - The pre-trained facial landmark detector inside the dib library is used to estimate the location of 68-(XXY) coordinates that map to the facial structure of the face [2]. These 68-(XXY) coordinates represent the important regions of the face like mouth, left eyebrow, right eyebrow, left eye, right eye, nose, and jaw. Of these, we only need the (XXY) coordinates of the left eye, right eye, and mouth.

This is done as:



Examining the image, we can see that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

The right eye using [36, 42] and the left eye with [42, 48].

Using the dictionary inside face_utils of the imutils library,we can easily extract the indexes into the facial landmarks array and extract various facial features simply by supplying a string as a key.

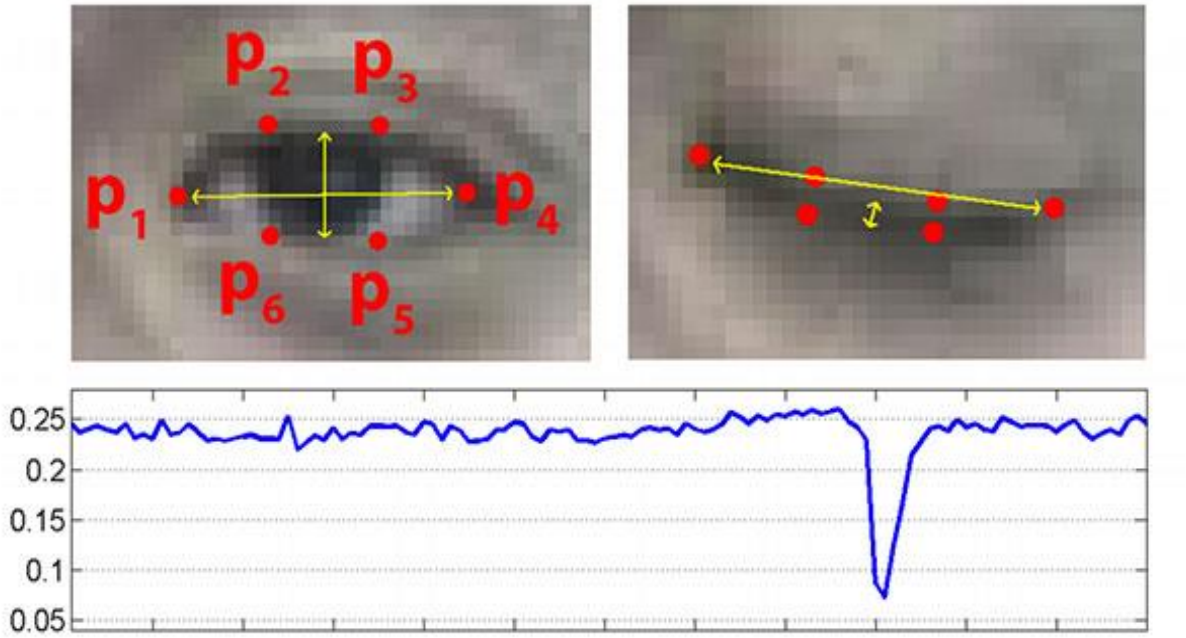
Therefore, to extract the eye regions from a set of facial landmarks, we simply need to know the correct array slice indexes:

Q2 - Describe techniques you would use to detect drowsiness in the eyes.

Ans- The drowsiness detector algorithm:

The general flow of our drowsiness detection algorithm is fairly straightforward.
First, we'll setup a camera that monitors a stream for faces:
If a face is found, we apply facial landmark detection and extract the eye regions:
If the eye aspect ratio indicates that the eyes have been closed for a sufficiently long enough amount of time, we'll sound an alarm to wake up the driver:

Function which is used to compute the ratio of distances between the vertical eye landmarks and the distances between the horizontal eye landmarks:
The return value of the eye aspect ratio will be approximately constant when the eye is open. The value will then rapidly decrease towards zero during a blink.
If the eye is closed, the eye aspect ratio will again remain approximately constant, but will be much smaller than the ratio when the eye is open.



On the top-left, we have an eye that is fully open — the eye aspect ratio here would be large(r) and relatively constant over time.
However, once the person blinks (top right) the eye aspect ratio decreases dramatically, approaching zero.
The bottom figure plots a graph of the eye aspect ratio over time for a video clip. As we can see, the eye aspect ratio is constant, then rapidly drops close to zero, then increases again, indicating a single blink has taken place.
As we can see, the eye aspect ratio is constant (indicating the eye is open), then rapidly drops to zero, then increases again, indicating a blink has taken place.
In our drowsiness detector case, we'll be monitoring the eye aspect ratio to see if the value *falls* but *does not increase again*, thus implying that the person has closed their eyes.

The eye aspect ratio (EAR) between height and width of the eye is computed.
There is a relation between the *width* and the *height* of these coordinates.

$$\mathbf{EAR} = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||}$$

Where *p1, …, p6* are 2D facial landmark locations.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only one set of horizontal points but two sets of vertical points.

The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

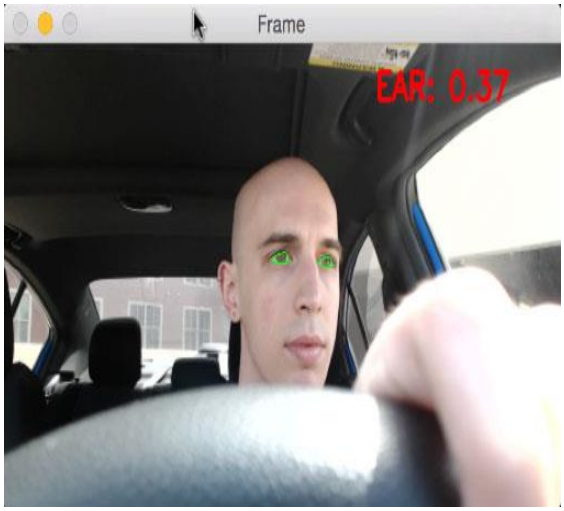
If the eye aspect ratio falls below this threshold, we'll start counting the number of frames the person has closed their eyes for.
If the number of frames the person has closed their eyes in exceeds, we'll sound an alarm.

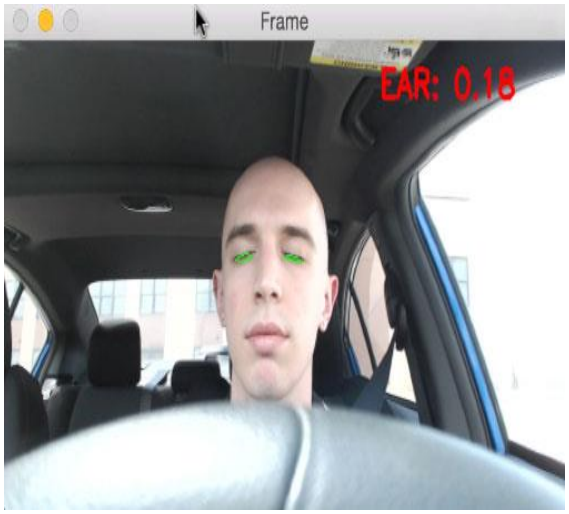
To start the core of our drowsiness detector:

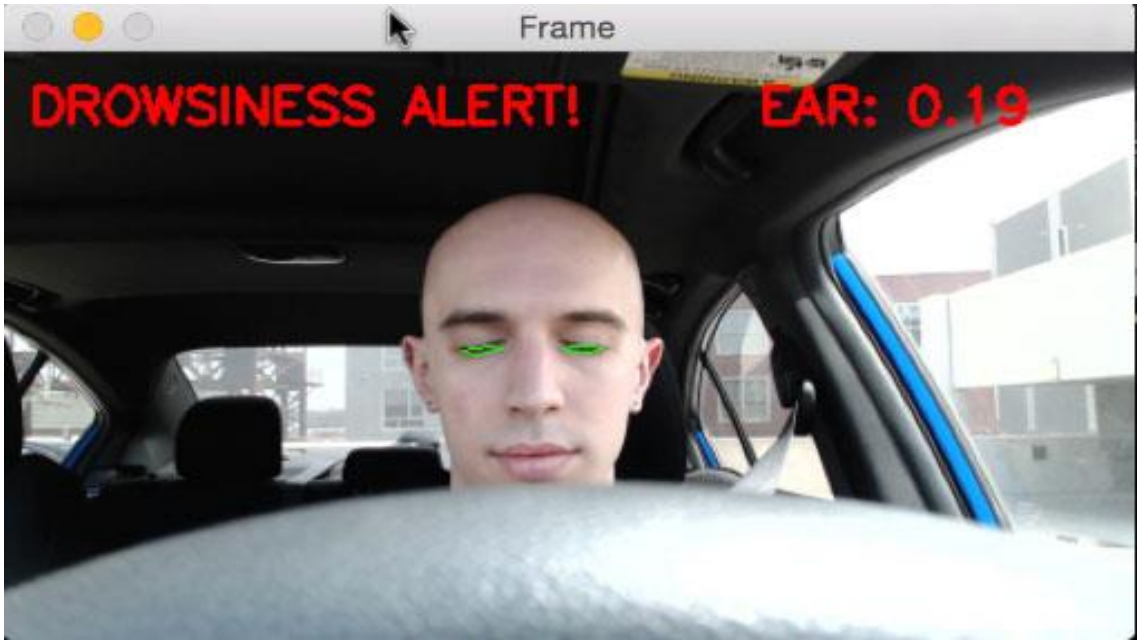
We instantiate our `VideoStream`using the supplied `--webcam` index.
We then pause for a second to allow the camera sensor to warm up.
We start looping over frames in our video stream.
Then we read the frame, which we then pre-process by resizing it to have a width of 450 pixels and converting it to grayscale.
Then we use dlib's face detector to find and locate the face(s) in the image.
The next step is to apply facial landmark detection to localize each of the important regions of the face:
The next step is to apply facial landmark detection to localize each of the important regions of the face: e loop over each of the detected faces on — in our implementation (specifically related to driver drowsiness), we assume there is only one face — the driver — but I left this `for` loop in here just in case you want to apply the technique to videos with more than one face.
For each of the detected faces, we apply dlib's facial landmark detector and convert the result to a NumPy array. Using NumPy array slicing we can extract the (x, y)-coordinates of the left and right eye, respectively. Given the (x, y)-coordinates for both eyes, we then compute their eye aspect ratios averaging both eye aspect ratios together to obtain a better estimation
We can then visualize each of the eye regions on our frame by using the cv2.drawContours function below this is often helpful when we are trying to debug our script and want to ensure that the eyes are being correctly detected and localized: We make a check to see if the eye aspect ratio is below the "blink/closed" eye threshold. If it is, we increment COUNTER, the total number of consecutive frames where the person has had their eyes closed.
If COUNTER exceeds frames then we assume the person is starting to doze off.
Another check is made, this time to see if the alarm is on — if it's not, we turn it on. Handle playing the alarm sound, provided a path was supplied when the script was executed. We take special care to create a separate thread responsible for calling sound alarm to ensure that our main program isn't blocked until the sound finishes playing.
Then we handle the case where the eye aspect ratio is larger than thresh, indicating the eyes are open. If the eyes are open, we reset COUNTER and ensure the alarm is off.

Q3-End the synopsis with a conclusion which should contain information about why you selected a particular algorithm over other existing solutions.

Conclusion - I could also use Adaboost Algorithm and Classifier Training. Adaboost algorithm is a kind of boosting algorithm proposed by Freund and Schapire , which selects some weak classifiers and integrates them into strong classifiers automatically. I selected this particular algorithm over other existing solutions because this includes OpenCV Library, Python, dib and some concepts which I am well versed with. As the results show, our drowsiness detector is able to detect when I'm at risk of dozing off and then plays a loud alarm to grab my attention. The drowsiness detector is even able to work in a variety of conditions, including direct sunlight when driving on the road and low/artificial lighting while in the concrete parking garage.







THANK YOU :)