Driver drowsy detection system

Tejal Mane
University of Michigan Dearborn, MI, USA
tmane@umich.edu

***Abstract*—Drivers often feel very drowsy while continuously driving for long distances without taking breaks. Drowsiness is a major factor in increasing the chances for a vehicle to meet accidents . According to various studies, Number of accidents caused by drowsiness is much higher than the number of accidents caused by drunk driving. The number of accidents caused by drowsiness can be reduced by having a proper system that can detect drowsiness, alert the driver and prevent major injuries. It needs a proper system that will alert drivers to prevent major injuries. The project starts an alarm when it detects the driver is drowsy and notify drivers to take a rest along with a nearby rest services area. It detects the driver's last sleeping time and suggests to take a rest**

***Index Terms*—Driver drowsiness, detection, driving safety, road accidents.**

I. Introduction

The increasing number of traffic accidents due to a driver's diminished vigilance level has become a serious problem for society. Some of these accidents are the result of the driver's medical condition [1]. However, a majority of these accidents are related to driver fatigue, drowsiness of drivers. Car accidents associated with driver fatigue are more likely to be serious, leading to serious injuries and deaths. Fletcher et al. has mentioned that 30% of all traffic accidents have been caused by drowsiness. It was demonstrated that driving performance deteriorates with increased drowsiness with resulting crashes constituting more than 20% of all vehicle accidents Traditionally transportation systems are no longer sufficient. One can use a number of different techniques for analyzing driver's drowsiness. These techniques are Image Processing based techniques, convolutional neural network based techniques. And image processing based techniques can be divided in three categories. These categories are template matching technique, eye blinking technique, yawning based technique [2]. These techniques are based on computer vision using image processing. In the computer vision technique, facial expressions of the driver like eyes blinking and head movements are generally used by the researchers to detect driver drowsiness.

A. Prerequisites

The requirement for this Python project is a webcam through which I will capture images. You need to have Python (3.6 version recommended) installed on your system, then using pip, you can install the necessary packages.

B. Dataset

The dataset used for this model is created by me. To create the dataset, I wrote a script that captures eyes from a camera and stores in our local disk. I separated them into their respective labels 'Open' or 'Closed'. The data was manually cleaned by removing the unwanted images which were not necessary for building the model. The data comprises around 7000 images of people's eyes under different lighting conditions. After training the model on our dataset, I have attached the final weights and model architecture file "models/cnnCat2.h5". Now, this model can be used to classify if a person's eye is open or closed.

C. Model Architecture

The model I used is built with Keras using Convolutional Neural Networks (CNN). A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple numbers of layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter. The CNN model architecture consists of the following layer

- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Fully connected layer; 128 nodes
- The final layer is also a fully connected layer with 2 nodes.

In all the layers, a Relu activation function is used except the output layer in which I used Softmax.

II. Project Design

A. Hardware and Software requirements

Hardware Required

- PC
- Webcam

Software Required

- OpenCV Face Eye Detection
- Keras – To build Classification Model
- Tensorflow-Keras uses TensorFlow as Backend
- Pygame-to play alarm sound

III. Implementation

A. Basic Workflow

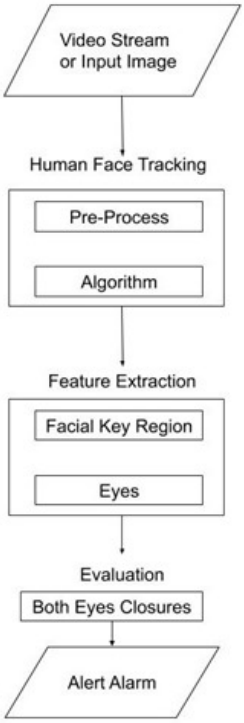

Fig. 1. Block Diagram of the system

B. Data Collection

A custom dataset of 7,000 images was created, capturing eye states under various lighting conditions. Images were labeled as "Open" or "Closed."

C. Algorithm

- Capture real-time images using a webcam.
- Detect facial regions and extract eye images.
- Process images (grayscale conversion, resizing, normalization).
- Predict eye state using the CNN model.
- Trigger alerts if drowsiness is detected

D. Algorithm Description with code snippets

The approach I will be using for this Python project is as follows [3].

- Step 1 – Take an image as input from a web camera. To access the webcam, I made an infinite loop that will capture each frame. I use the method provided by OpenCV, cv2.VideoCapture(0) to access the camera and set the capture object (cap). cap.read() will read each frame and I store the image in a frame variable.
- Step 2 – Detect the face in the image and create a Region of Interest (ROI). To detect the face in the image, I need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. I don't need color information to detect the objects. I will be using a haar cascade classifier to detect faces. This line is used to set our classifier: face = cv2.CascadeClassifier('path to our haar cascade XML file'). Then I perform the detection using faces = face.detectMultiScale(gray). It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now I can iterate over the faces and draw boundary boxes for each face. Code Snippet: 1. for (x,y,w,h) in faces: 2. cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1)
- Step 3 – Detect the eyes from ROI and feed it to the classifier. The same procedure to detect faces is used to detect eyes. First, I set the cascade classifier for eyes in leye and reye respectively then detect the eyes using left_eye = leye.detectMultiScale(gray). Now I need to extract only the eye data from the full image. This can be achieved by extracting the boundary box of the eye and then I can pull out the eye image from the frame with this code. Code Snippet: l_eye = frame[y : y+h, x : x+w] l_eye only contains the image data of the eye. This will be fed into our CNN classifier which will predict if eyes are open or closed. Similarly, I will be extracting the right eye into r_eye.
- Step 4 – The classifier will categorize whether eyes are open or closed. We are using CNN classifiers for predicting eye status. To feed our image into the model, I need to perform certain operations because the model needs the correct dimensions to start with. First, I convert the color image into grayscale using r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY). Then, I resize the image to 24*24 pixels as our model was trained on 24* 24-pixel images cv2.resize(r_eye, (24,24)). I normalize the data for better convergence r_eye = r_eye/255 (All values will be between 0-1). Expand the dimensions to feed into our classifier. I loaded our model using model = load_model('models/cnnCat2.h5'). Now we predict each eye with our model lpred = model.predict_classes(l_eye). If the value of lpred[0] = 1, it states that eyes are open, if value of lpred[0] = 0 then, it states that eyes are closed.
- Step 5 – Calculate the score to check whether the person is drowsy. The score is a value I will use to determine how long the person has closed his eyes. So if both eyes are closed, I will keep on increasing the score and when eyes are open, we decrease the score. I are drawing the result on the screen using the cv2.putText() function which will display the real-time status of the person. Code Snippet : cv2.putText(frame, "Open", (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA) A threshold is defined for example if score becomes greater than 15 that means the person's eyes are closed for a long period of time. This is when I beep the alarm using sound.play()

IV. Results

The system successfully detects drowsiness with high accuracy under various lighting conditions and for drivers wearing spectacles. Prolonged eye closure triggers alarms, effectively reducing the risk of accidents.

A. Output

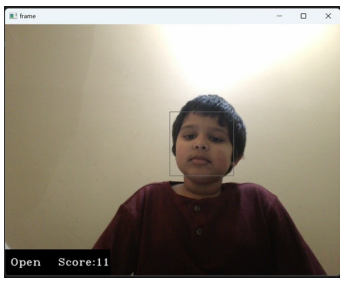

Fig. 2. Output heuristics when eyes are open

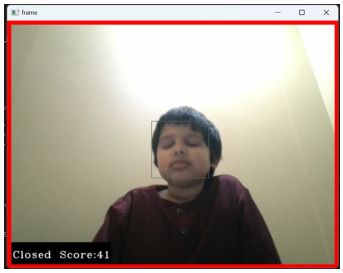

Fig. 3. Output heuristics when eyes are closed

V. Conclusion and Future Aspects

The Accident Reduction Aid system developed is capable of detecting drowsiness in a rapid manner. The system which can differentiate normal eye blink and drowsiness which can prevent the driver from entering the state of sleepiness while driving. The system works well even in case of drivers wearing spectacles and under low light conditions also. During monitoring, the system can decide whether the eyes are open or closed. When the eyes have been closed for about two seconds, the alarm beeps to alert the driver. By doing this many accidents will be reduced and provides safe life to the driver and vehicle safety.

The future works may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc, for fatigue measurement. Driver drowsiness poses a major threat to highway safety, and the problem is particularly severe for commercial motor vehicle operators. Twenty-four-hour operations, high annual mileage, exposure to challenging environmental conditions, and demanding work schedules all contribute to this serious safety issue. Monitoring the driver's state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem. Currently there is not adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized.

References

[1] V. Saini and R. Saini, "Driver drowsiness detection system and techniques: a review," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4245–4249, 2014.

[2] R. O. Mbouna, S. G. Kong, and M.-G. Chun, "Visual analysis of eye state and head pose for driver alertness monitoring," *IEEE transactions on intelligent transportation systems*, vol. 14, no. 3, pp. 1462–1469, 2013.

[3] H. Singh, J. S. Bhatia, and J. Kaur, "Eye tracking based driver fatigue monitoring and warning system," in *India International Conference on Power Electronics 2010 (IICPE2010)*. IEEE, 2011, pp. 1–6.