Driver Drowsiness Detection using Machine Learning

Bala Naga Sai Venkata Bharath Manda
19991112-T193
bamn20@student.bth.se
Department Of Computer Science
Blekinge Institute Of Technology

***Index Terms*—Machine Learning, Convolution Neural Networks(CNN), Drowsiness Detection**

I. Introduction

This report explains the approach used to detect the level of drowsiness in a driver. Estimates show that around 91,000 accidents involved drowsy drivers. These crashes led to an estimated 50,000 people being injured and nearly 800 deaths[10]. These numbers show us how crucial it is to have detection systems. The model works on the basis of the driver's eyes, i.e constantly monitoring the region of interest to classify whether the driver is drowsy or not. If the driver tends to be drowsy, the system intelligently alerts the driver to avoid any possible accidents. The dataset used for training the model was obtained from Kaggle[9] namely 'yawn_eye_dataset_new' with 7000 instances. The model was trained using the HAAR Cascade algorithm which proved to be a very efficient and fast algorithm for OpenCV involving facial features such as eyes, nose, mouth etc.

II. Related Work

Malla et al. [1] developed a light-insensitive system. They used Haar algorithm to detect various objects [2] and face classifier which is implemented by [3] in OpenCV [4] libraries. They derived the eye regions from the facial region. Then, the eyelid is detected which measure the level of eye closure.
Based on an infrared camera, Vitabile et al. [5] developed a method to identify indications of driver drowsiness. An algorithm for identifying and monitoring the driver's eyes has been developed using the phenomena of bright pupils. When the technology detects sleepiness, an alarm message is sent to the driver.
Bhowmick et Kumar [6] utilized the Otsu thresholding [7] to extract face area. The localization of the eye is done by locating facial landmarks such as eyebrow and possible face center. For precise eye segmentation, morphological operations and K-means are applied. Then a set of shape features are computed and trained using non-linear SVM to determine the status of the eye.
Hong et al. [8] proposed a system for detecting driver drowsiness by sensing eye states in real time. The improved Jones and Viola approach [2] is used to detect the face region. An horizontal projection is used to acquire the eye area. Finally, to detect the eye state, a new complexity function with a dynamic threshold was developed.
Tian et Qin [9] built a system that monitors the driver's eye states. The Cb and Cr components of the YCbCr color space are used in their system. This technique uses a vertical projection function to locate the face and a horizontal projection function to locate the eyes. Once the eyes have been found, the system uses a function of complexity to determine the states of the eyes.
Under the light of what has been mentioned above, the identification of the driver drowsy state given by the PERCLOS is generally passed by the following stages:
-Face detection
-Eyes Location
-Face and eyes tracking
-Identification of the eyes states
-Calculation of PERCLOS and identification of driver state

III. Method Implementation

The main aim of this project is to build a machine learning model that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected. To implement this, I will be using OpenCV for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed'. The steps for implementing this system are as follows:
Step 1 – Take image as an input from camera.
Step 2 – Detect the face in the image and create a Region of Interest(ROI).
Step 3 – Detect the eyes from ROI and feed it to the classifier.
Step 4 – Classifier will categorize whether the eyes are open or closed.
Step 5 – Calculate score to check whether the person is drowsy.

A. Convolution Neural Networks(CNN)

The model that I'd like to implement is Convolutional Neural Networks (CNN). The model I've 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 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 layers:
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. A Relu activation function is used in all the layers except the output layer in which i've used Softmax.

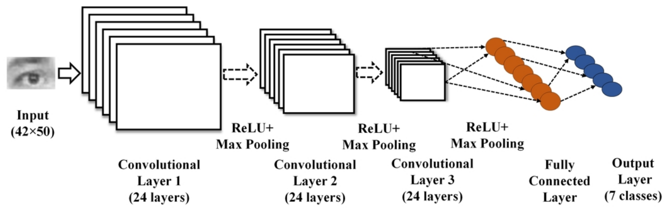


Fig. 1. Convolution Neural Networks Architecture

B. Dataset

The dataset for this project was obtained from Kaggle. The data comprises around 7000 images of people's eyes under different lighting conditions. After training the model by using this dataset, I have generated a model file for the model with final weights and saved the model file in "models/CNNFinal.h5" this folder.
Now, you can use this model to classify if a person's eye is open or closed.

C. Implementation

Step 1 – Take image as an input from camera
By using a webcam, images are taken as an input. So for accessing the webcam, I made an infinite loop that will capture each frame. I have used method provided by OpenCV that is "cv2.VideoCapture(0)" to access the camera and set the capture object (cap). cap.read() will read each frame and stores 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, the image need to be converted into grayscale because OpenCV algorithm for object detection takes only gray images as input. The color information is not required to detect the objects. And I have used haar cascade classifier to detect faces. This line in the code is used to set our classifier face = cv2.CascadeClassifier(' path to particular haar cascade xml file'). Then performed 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 iterate over the faces and draw boundary boxes for each face. Step 3 – Detect the eyes

```
for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )
```

from ROI and feed it to the classifier
The process for detecting eyes is the same as the process for detecting faces. First, set the cascade classifier for eyes in leye and reye respectively. Then detect the eyes using left_eye = leye.detectMultiScale(gray). Now extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then pulling out the eye image from the frame with this code.
l_eye only contains the image data of the left eye. Similarly, we will be extracting the right eye into r_eye. Then, this will be fed into CNN classifier which will predict if eyes are open or closed. Step 4 – Classifier will categorize whether the eyes

```
for (x,y,w,h) in left_eye:
    l_eye=frame[y:y+h,x:x+w]
```

are open or closed
We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with. First, we convert the color image into grayscale using r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY). Then, we resize the image to 24*24 pixels as our model was trained on 24*24 pixel images cv2.resize(r_eye, (24,24)).Next, normalize our data for better convergence r_eye = r_eye/255 (All values will be between 0-1). Expand the dimensions to be fed into the classifier. The model is loaded using model = load_model('models/CNNFinal.h5'). Now, the model will predict each eye by
lpredict=model.predict(l_eye)
lpred=np.argmax(lpredict,axis=1)
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 score to check whether the person is drowsy
The score is just a number that we'll use to figure out how long the person has been closed-eyed. As a result, if both eyes are closed, we will continue to increase the score, however if both eyes are open, we will drop the score. The result is displayed on the screen using cv2.putText() function which will display real time status of the person. cv2.putText(frame, "Open", (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA)
A threshold can be defined like if score becomes greater than 15 that means the person's eyes are closed for a long period of time. If the threshold is reached then beep the alarm using sound.play().

```
if(rpred[0]==0 and lpred[0]==0):
    score=score+1
    cv2.putText(frame,"Closed",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
#if(rpred[0]==1 or lpred[0]==1):
else:
    score=score-1
    cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
```

IV. Results

The results generated contain the images of a person with eyes open and eyes closed. If the eyes are closed with a score of 15 or more i.e, the threshold amount, the model produces an alarm to alert the driver. The metric used for evaluating the model was accuracy_score.

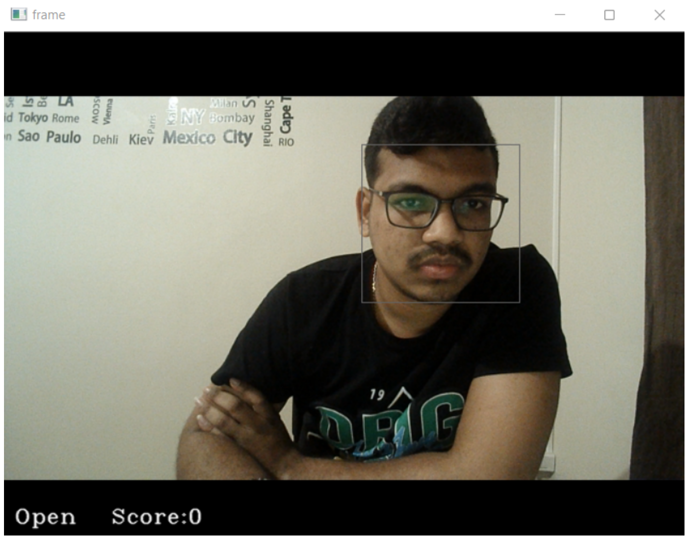


Fig. 2. Eyes opened image

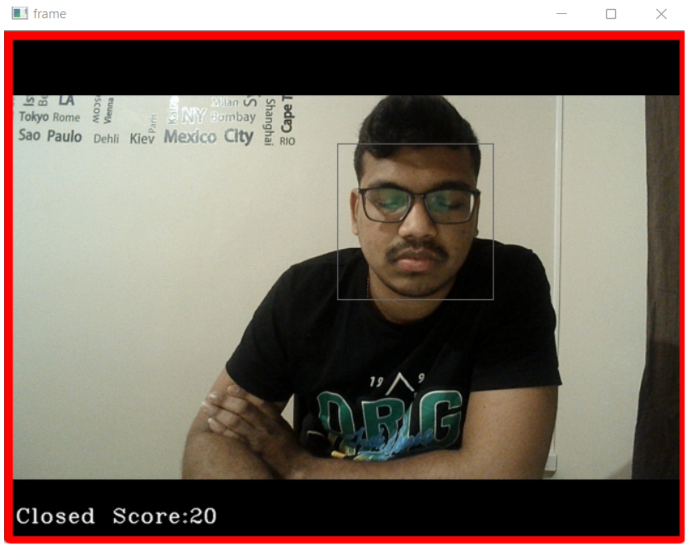
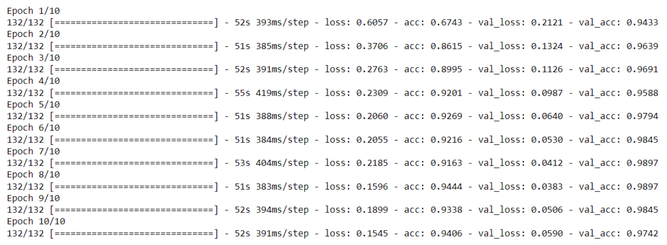


Fig. 3. Eyes closed image

A. Accuracy score

It is an evaluation metric used for summarizing the performance of a classification model. Higher the accuracy, better the performance of the model. With my model i was able to achieve an accuracy of 97%.



```
Epoch 1/10
132/132 [==============================] - 52s 393ms/step - loss: 0.6057 - acc: 0.6743 - val_loss: 0.2121 - val_acc: 0.9433
Epoch 2/10
132/132 [==============================] - 51s 385ms/step - loss: 0.3706 - acc: 0.8615 - val_loss: 0.1324 - val_acc: 0.9639
Epoch 3/10
132/132 [==============================] - 52s 391ms/step - loss: 0.2763 - acc: 0.8995 - val_loss: 0.1126 - val_acc: 0.9691
Epoch 4/10
132/132 [==============================] - 55s 419ms/step - loss: 0.2309 - acc: 0.9201 - val_loss: 0.0987 - val_acc: 0.9588
Epoch 5/10
132/132 [==============================] - 51s 388ms/step - loss: 0.2060 - acc: 0.9269 - val_loss: 0.0640 - val_acc: 0.9794
Epoch 6/10
132/132 [==============================] - 51s 384ms/step - loss: 0.2055 - acc: 0.9216 - val_loss: 0.0530 - val_acc: 0.9845
Epoch 7/10
132/132 [==============================] - 53s 404ms/step - loss: 0.2185 - acc: 0.9163 - val_loss: 0.0412 - val_acc: 0.9897
Epoch 8/10
132/132 [==============================] - 51s 383ms/step - loss: 0.1596 - acc: 0.9444 - val_loss: 0.0383 - val_acc: 0.9897
Epoch 9/10
132/132 [==============================] - 52s 394ms/step - loss: 0.1899 - acc: 0.9338 - val_loss: 0.0506 - val_acc: 0.9845
Epoch 10/10
132/132 [==============================] - 52s 391ms/step - loss: 0.1545 - acc: 0.9406 - val_loss: 0.0590 - val_acc: 0.9742
```

Fig. 4. Accuracy

V. Conclusion

In this report, I have implemented a drowsy driver alert system that can be applied in various fields. I have used OpenCV to detect faces and eyes using a Haar cascade classifier which complements the Convolutional Neural Network model to predict the status of the eye. The evaluation metric used in this report is the Accuracy score and was able to achieve a high accuracy score of 97.4% with the yawn_eye dataset.

References

[1] P. B. R. G. A. Malla, P. Davidson and R. Jones, “Automated video-based measurement of eye closure for detecting behavioral microsleep,” 32nd Annual International Conference of the IEEE, Buenos Aires, Argentina, 2010.

[2] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[3] R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection,” Proceedings of the IEEE International Conference on Image Processing, 2002.

[4] “Opencv, open source computer vision library reference manual,” 2001.

[5] A. P. S. Vitabile and F. Sorbello, “Bright pupil detection in an embedded, real-time drowsiness monitoring system,” 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[6] B. Bhowmick and C. Kumar, “Detection and classification of eye state in ir camera for driver drowsiness identification,” Proceeding of the IEEE International Conference on Signal and Image Processing Applications, 2009.

[7] N. Otsu, “A threshold selection method from gray-level histograms,” IEEE Transactions on Systems,Man and Cybernetics, 1979.

[8] H. Q. T. Hong and Q. Sun, “An improved real time eye state identification system in driver drowsiness detection,” IEEE International Conference on Control and Automation, Guangzhou, CHINA, 2007.

[9] Serenraju, “yawn_eye_dataset_new for drowsiness detection,”

[10] Drowsy-Driving, “https://www.nhtsa.gov/risky-driving/drowsy-driving,”