Final Project Report – Driver Fatigue Detection

ID2223 HT20-1 Scalable Machine and Deep Learning

Yang Nan (nanyang@kth.se) & Varrun Varatharajan (varrun@kth.se)

Table of Contents

1. Introduction **1**
2. Dataset **2**
3. Method **2**
   a. Data Pre-processing 2
   b. Classification Model 2
   c. Summary of steps 2
4. Results **3**
5. Code Instructions **5**
   Resources **5**

1. Introduction

Somewhere between 20-24% of accidents on the road happen because of exhausted, sleepy drivers. This endangers their own lives as well as that of the pedestrians and other motorists. [1] As students of autonomous systems, we strive to make cars not only autonomous, but also the roads safer. Our goal in this project is to create a real-time fatigue detection system using deep learning networks that are trained on the driver's eye.

2. Dataset

Finding large datasets with multiple images of unique faces is a tedious task. Hence, the model was trained on a large dataset found on GitHub repositories. The dataset consists of 2425 low resolution images of well-known and random personalities taken from stock images on the internet. It consists of 1232 images with eyes open, and 1193 images with eyes closed. The dataset is attached in the submission as a zip file.

Finally, the model was validated by using a video from personal webcams.

3. Method

a. Data Pre-processing

The tools used for the analysis and visualization of data are: OpenCV, Tensorflow, Keras, and Google Colab notebook.

Of the 2423 images, 1939 were used for training and 484 used for validation. Class labels were created which are based on faces with open and closed eyes named ‘ClosedFace', 'OpenFace' respectively. The haar cascades consist of xml files that contain objects that are essential to detect faces and eyes in the image. This forms the basis for the classification model to then train on the detected portions of eyes that are labelled as either closed or open.

b. Classification Model

The deep learning model built using Keras is a Convolutional Neural Network (CNN). This model was chosen because of its well-known performance for classification of images. The CNN has input, hidden and output layers, with a 2D filter that performs matrix multiplication on the layers.

The architecture of the CNN is as described:-

- Convolutional layer= 16 nodes, Kernel= 3, Activation= ReLU
- Convolutional layer= 32 nodes, Kernel= 3, Activation= ReLU
- Convolutional layer= 64 nodes, Kernel= 3, Activation= ReLU
- Convolutional layer= 128 nodes, Kernel= 3, Activation= ReLU
- Convolutional layer= 256 nodes, Kernel= 3, Activation= ReLU
- Fully Connected layer= 1024 nodes
- Fully Connected layer= 128 nodes
- Fully Connected layer= 32 nodes
- Data Augmentation layer
- Dropout layers

c. Summary of steps

Webcam image input

cv2.VideoCapture() in OpenCV method helps access webcam and store it as an object frame-by-frame. There are a total of 466 frames in our video.

Face detection Region of Interest (ROI)

OpenCV's haarcascade only takes black and white frames as input. Hence, the frames are converted such.

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_alt.xml')
```

`faces = face_cascade.detectMultiScale` helps detect the face ROI. `frame[y:y+h, x:x+w, :]` fixes the bounding boxes for faces. It is iterated so that the same is done over all frame objects. `layers.experimental.preprocessing.Rescaling(1./255)` is then done to normalize data for better convergence. The values then all fall between 0 to 1.

CNN classifies whether eyes are Open and Closed

`cv2.cvtColor()` is used to convert the images into grayscale, then `cv2.resize` is used to resize the frames to match the webcam frames to the size of training image datasets.

Calculate drowsiness score based on how long the eyes are closed

The score is determined based on how long the eyes are open or closed. This should not be confused with blinking, hence, if the eyes are closed for less than 0.5 seconds, nothing happens and the system indicates that 'You are awake'. However, if that period of eye shut is between 0.5 to 1 second, the counter increases and a 'Warning' appears. This counter keeps increasing as long as the eyes remain closed. If the eyes remain closed beyond 1 second, the system recognizes that the person is asleep, and indicates 'Wake Up'.

As a real-time practical extension, this can further be connected to a physical indicator or alarm in the vehicle to beep once the threshold is crossed, hence proving the project to be of great use for the automotive industry.

4. Results

The resulting file of the classifier has been shared as a video file with the project.

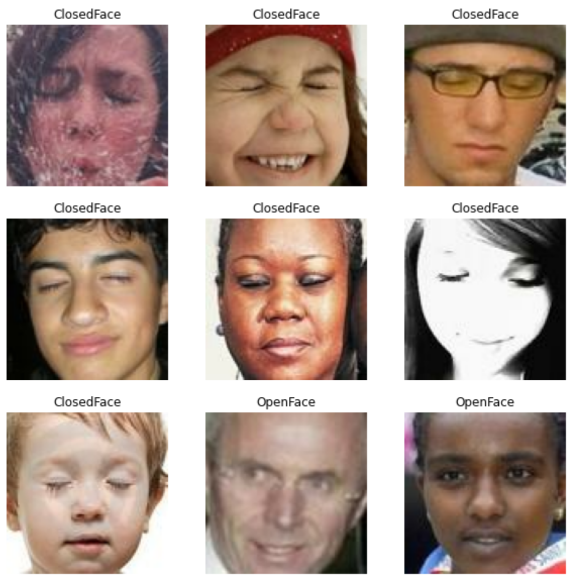

Figure 1. Manually classified datasets of open and closed eyes

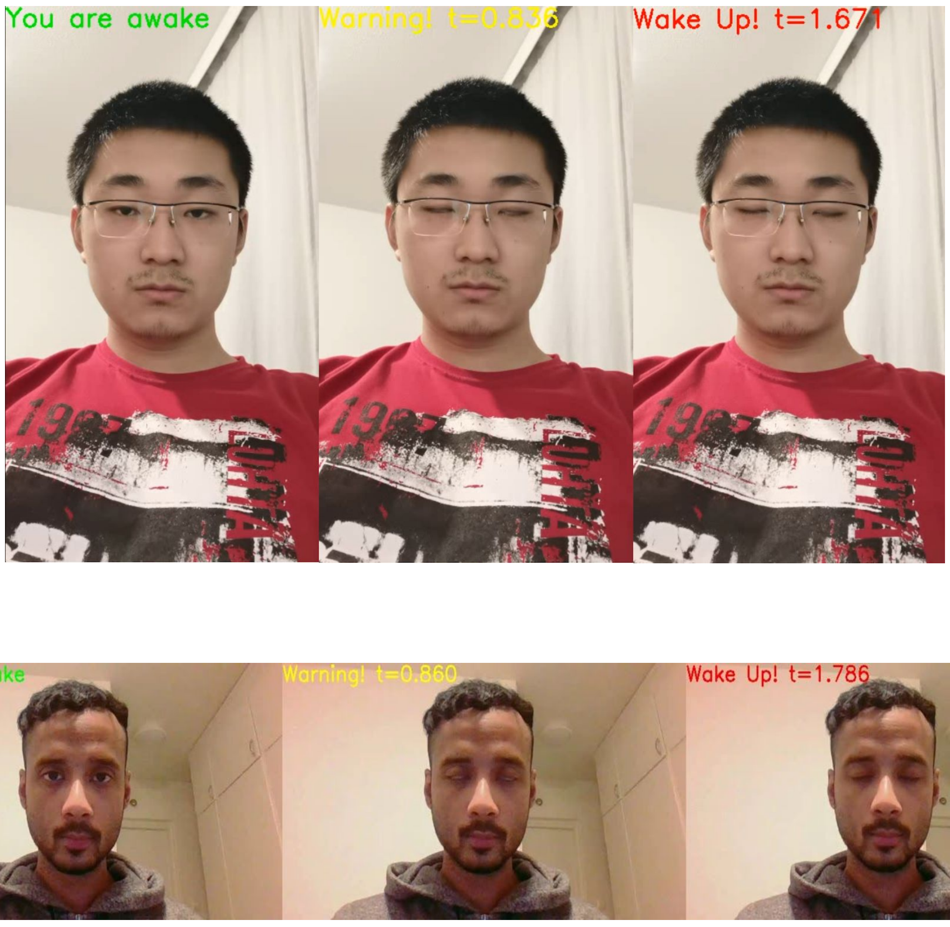

Figure 2. Test results on webcam

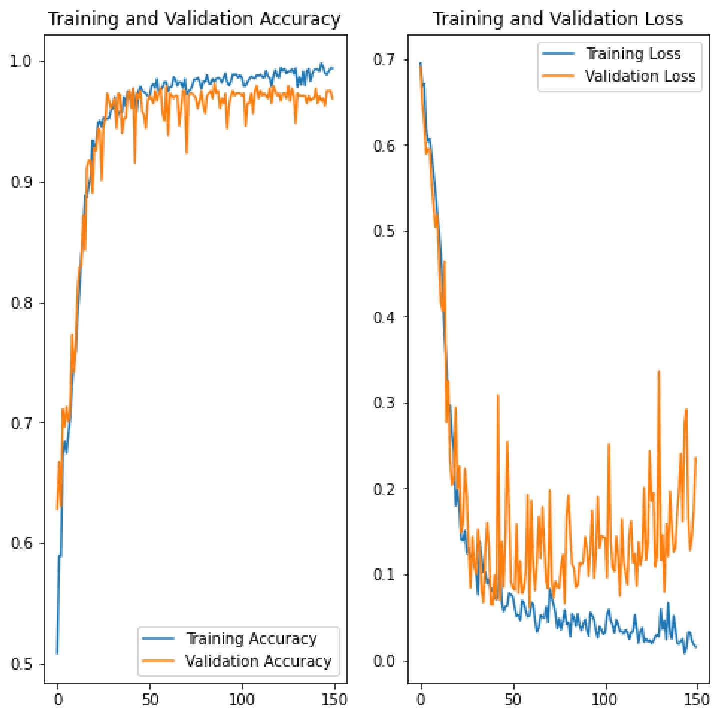

Figure 3. Plots for Training and Validation Accuracy and Loss

The Training and Validation Loss plots show a classic case of overfitting. The curves appear to stabilize and merge below 50 epochs, after which it starts diverging again. This could happen if the model has more capacity than required for this scenario. The learning stops happening after the inflection point. [2]

Overfitting here is happening due to too little data to train the model. Increasing the training dataset mitigates this problem. We use data augmentation in the project to solve this problem. [3] Data Augmentation uses random examples from the existing dataset by providing realistic transformations. This leads the model to train on more data in different perspectives.

5. Code Instructions

The code can be found in the file drowsiness_detection.ipynb. The zip file also contains the dataset data.zip, and the resulting video files result_var.avi, result_yang.avi.

To replicate the results of your own, upload a video of closing eyes using recorded webcam video. Upload the link of the video location in the 'origin' under the Download Video section of the notebook. Change the input filenames as required and run the notebook. The output result.avi can be found on the left tab under 'Files'.

Resources

[1]https://www.tac.vic.gov.au/road-safety/statistics/summaries/fatigue-statistics

[2]https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance

[3]https://www.tensorflow.org/tutorials/images/classification#data_augmentation