Drowsiness Detection for Safe Driving Using PERCLOS and YOLOv2 Method

Ngo Kah Lock, Weng Mun Ng, Norrabiatul Adawiyah Jusoh, Nazhatul Hafizah Kamarudin, Rizauddin Ramli, and Zuliani Zulkoffli

Abstract Driver drowsiness is considered as a significant risk factor that impacted to large number of road accidents. This paper presents a vehicle safety system developed using computer vision technique to detect driver drowsiness while driving. Deep learning method, YOLOv2 has been chosen as the artificial intelligence model to detect the percentage of eyes closure of the driver. A threshold value for percentage eye closure for the driver, where is when the set threshold value of 20% has been reached, response from a connected buzzer will alert the driver. This system development involves a few stages of dataset preparation and training. An accuracy of 86.58% has been reached to detect eyes closure in this training model. The trained model had been simulated using NVIDIA Jetson Nano platform to simulate real world application and find out a few areas of improvement such as the frequency of yawning additional feature parameter to detect drowsiness and self-training mode for face recognition and eyes dataset of the driver was recommended to personalize the development application and minimizing inaccurate output.

Keywords Computer vision · Autonomous vehicle technology · Drowsiness detection · Jetson nano · PERCLOS and YOLO

1 Introduction

Fatigue and drowsiness driving has been identified as primary reasons for road accidents, especially when driving in the long-distance and monotonous motorway [1–5].

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-981-19-1577-2_9.

N. K. Lock · W. M. Ng · N. A. Jusoh · N. H. Kamarudin · Z. Zulkoffli (✉)
Faculty of Engineering, Technology and Built Environment, UCSI University, 56000 Kuala Lumpur, Malaysia
e-mail: Zuliani@ucsiuniversity.edu.my

R. Ramli
Department of Mechanical and Manufacturing Engineering, Universiti Kebangsaan Malaysia, UKM, 43600 Bangi, Selangor Darul Ehsan, Malaysia

Under these unwanted situations, it could become very dangerous for an individual to drive on the road, especially when the driver is not aware of his or her consciousness at that particular moment. However, although most individuals are aware of these significant issues, under such circumstances where drivers cannot keep themselves awake due to tiredness from daily activities, it is hard for drivers to stay focus and not fall asleep while driving on the road.

There are several approaches that were proposed to measure the driver drowsiness while driving. These approaches can be categorized into three main categories which included physiological approach, vehicle-based approach and behavioral metrics [6, 7]. The physiological and vehicle-based approach were a contact-based method that needs to attach sensors to the vehicle or the driver while the behavioral based approach was a contactless based method [8–10]. In addition, behavioral based approach can be divided into three sub-categories, which included appearance-based, template-based, and feature-based methods [11, 12]. Drowsiness detection using physiological approach is highly relying on the instrument, and it is not applicable for real-time detection and the vehicle-based approach is not reliable as it tends to get different results in a different environment. As a result, the behavioral approach will be used in the application to detect the drowsiness level of the driver. Percentage of eyelid closure (PERCLOS) over the pupil over time is the most often utilized method or technology in terms of measuring the drowsiness of the driver under the behavioral approach. To solve the problems of real time detection, low detection accuracy, and slow processing speed, a new PERCLOS detection model using YOLOv2 is proposed in this paper [13–16].

2 Methodology

This system development consists of two stages of programming part and testing part which is embedded in Jetson Nano. For programming part, the flowchart development for this drowsiness measurement was shown in Fig. 1. The system starts when the camera automatically turns on and captures images with a maximum rate of 30 frames per second. After that image acquisition process, the deep learning object detector, YOL0v2 will detect the regional proposals and classify the interest object of eyes simultaneously. It is set the toggle will be "l" if the eyes are detected in a closing state, and toggle will be "0" if the eyes are detected in an opening state.

The internal system detects the vector results of 5 s with 30 frames per seconds. The system will be undergone preprocessing stage of deleting the first array, which is zero array vector if the array size is bigger than 150 and save the toggle value into the last array. PERCLOS percentage was obtained using the sum of the toggle value in the array divided by 150, which equals the total number of the array and multiply by 100. If the PERCLOS is larger than 20%, then the driver is considered as "sleepy" whereas if the "PERCLOS" is lower or equal to 20%, then the driver is considered as "awake". The system will shut down once the "Q" button is pressed, or else the system will loop again to detect regional proposal and classify the object.

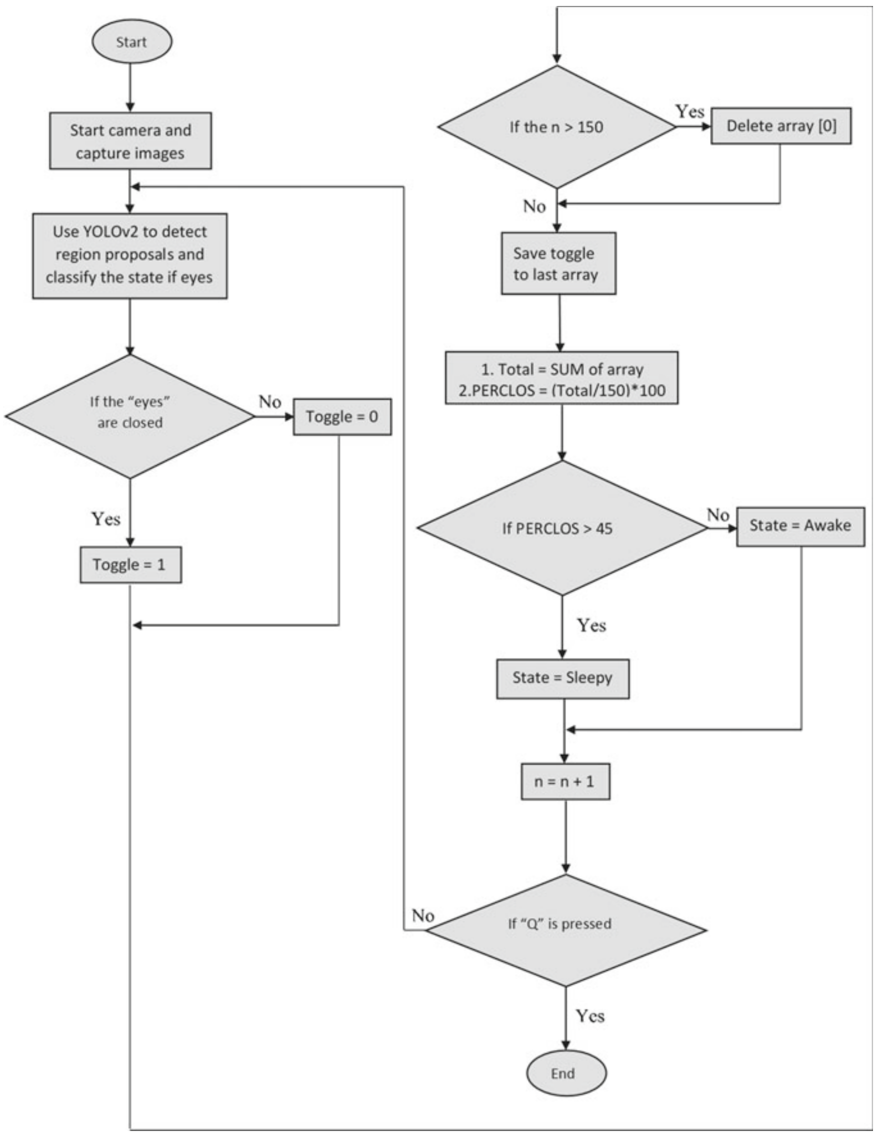


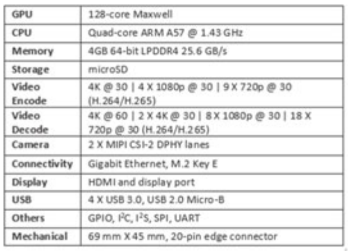
Fig. 1 Flowchart of PERCLOS detection

2.1 Hardware Setup

The device used to measure the drowsiness of the driver included a NVIDIA Jetson Nano microprocessor to process the input and generate output, a camera to capture images as input to NVIDIA Jetson Nano and a buzzer to provide response alert to the driver. Figure 2(a) shows the technical specifications of NVIDIA Jetson Nano used in this development. Figure 2(b) shows the schematic hardware setup of Jetson Nano with triple connections. Below is the procedure of hardware connections:

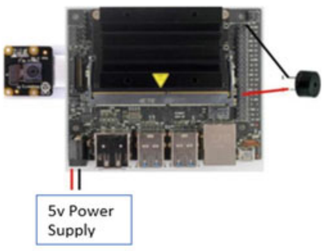
i) The connection of 5 V input voltage parallel with the car radio,
ii) Pi camera connect to the camera socket of the NVIDIA Jetson Nano using a ribbon cable
iii) Buzzer to alert the driver will connect to GPIO pins of the Jetson Nano, which positive terminal will connect to pin 12 of the GPIO pins and negative terminal connected to the ground pin.

As this device will be installed in car, devices housings for protection to this device is needed and made it able to fix in car. The devices housing has been designed using Solidworks software and printed using 3D Printer. Figure 2(c) shows the illustration of devices housing using Solidworks software.

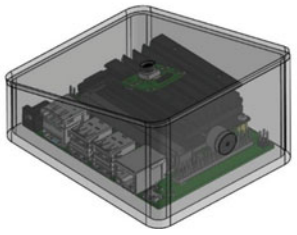


GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD
Video Encode	4K @ 30 | 4 X 1080p @ 30 | 9 X 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2 X 4K @ 30 | 8 X 1080p @ 30 | 18 X 720p @ 30 (H.264/H.265)
Camera	2 X MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4 X USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69 mm X 45 mm, 20-pin edge connector

(a)



(b)



(c)

Fig. 2 **a** Jetson Nano specifications **b** Schematic diagram for Jetson Nano hardware setup and **c** Illustrations of device's housing sketched using solidworks software

2.2 Algorithm Development of Eye Closure Detection Using Deep Learning Technique

YOL0v2 was chosen for this system development using deep learning model pre-trained on an architecture which is called Darknet-19. The structure of Darknet-19 is consisting of 19 convolutional layers and five maxpooling layers. Each Convolution layer has the BatchNorm normalization and then Leaky ReLu activation except for the last Convolution block. The Darknet-19 was initially trained with 1000 classes of classification dataset which is at 224×224 pixel^2 from standard ImageNet.

2.3 Data Acquisition for Deep Learning's Model Training

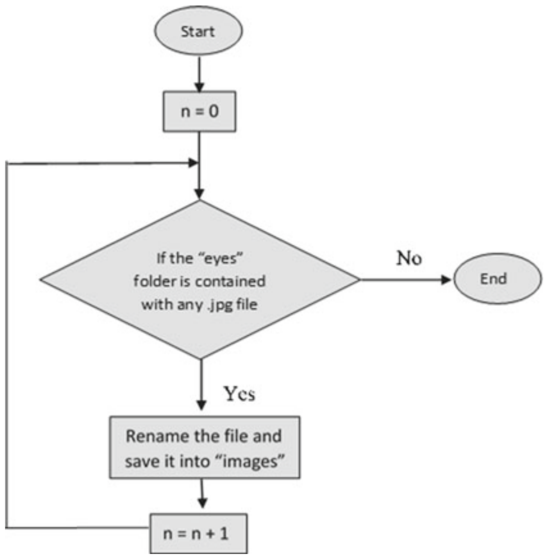
In this development to detect the drowsiness level of the driver, closed eyes in the wild (CEW) dataset and KomNet dataset were combined and used in deep learning model training. CEW dataset contains 2423 images in total, including 1192 images with close eyes and 1231 images with open eyes, collected directly over internet. Meanwhile, the KomNet dataset consists of 50 persons and 24 images each, a total of 1200 images. The collected files were in the format of.jpg. Face images were taken with a frontal face facing the camera.

2.4 Data Pre-processing

Datasets were prepared and generated the annotation files for the computer to recognize before performing training on deep learning's model. Beforehand, a python script code to rename the images from 0 to the total number of dataset images. This code allows the other python script to read the images more easily to generate annotate file. Next, a "data labelling" python script code is sketched to display the images, where it allows the user to use the rectangular selector to select the object. Then, the key binding is used to classify the classes of objects and the "generate.xml" python script is sketched in order to generate annotation files for training purpose. In this application, the images need to be classified into two classes, which are the opened eyes class and the close eyes class. Figure 3 shows the flowchart of the "data labelling" and "generate.xml" in python script.

Heretofore, there is no complete dataset for open eyes and close eyes, all the images downloaded from CEW dataset or KomNet dataset need to go through pre-processing stage before training stage. Pre-processing stage essential to align the dataset information and maximize the performance of the model. Example of open eyes data is shown in Fig. 4(a), which is circled with a green ellipse while close eyes data in Fig. 4(b), which is circled with a red ellipse.

Fig. 3 Flowchart for "data labelling" and "generate.xml files"


Start
n = 0
If the "eyes" folder is contained with any .jpg file
No
End
Yes
Rename the file and save it into "images"
n = n + 1

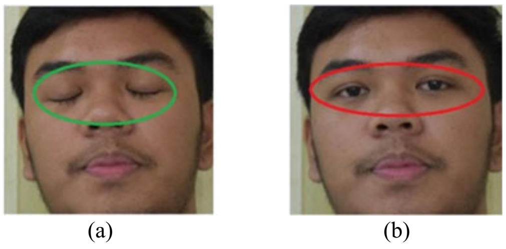


Fig. 4 Dataset examples **a** closed eyes and **b** open eyes

2.5 Deep Learning's Model's Training and Fine-Tuning

To train the model detecting open eyes and closed eyes, the pre-trained model must be obtained. It is to get the pre trained image feature extractor process to infer its bounding boxes. The pre-trained weights of the Darknet 19 was downloaded from the original YOL0v2 project website [13, 14]. At this point, with the complete pre-trained model, images dataset and annotations files, the model is ready to train.

Multi-scale training using YOL0v2 that is robust to run on different input images sizes, since the model uses only convolutional and pooling layers was chosen in this project development. YOL0v2 architecture changes the network every few iterations and randomly choose a new image dimension size from the dimension set. This makes the network able to predict the object at different resolutions. Neural networks use optimizing strategies like stochastic gradient descent to minimize the error in the algorithm and loss function is used to compute this error, to quantify how good or bad the model is performing [14]. YOL0v2 is composed of 3 losses, including confidence loss, localization loss and classification loss. Therefore, the losses are calculated and combined [16] as shown in Eq. 1.

Confidence Loss

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}(C_i - \hat{C}_j)^B + \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{no\,obj}(C_i - \hat{C}_j)^2$$

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left[(x_i - \hat{x}_j)^2 + (y_i - \hat{y}_j)^2\right]$$

Localization Loss

$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2\right]$$

Classification Loss

$$\sum_{i=0}^{S^2} 1_{ij}^{obj} \quad \sum_{c \in lasses} (pi(c) - \hat{p}i(c))^2$$

$$\text{Losses} = \text{Confidence loss} + \text{Localization loss} + \text{Classification loss} \quad (1)$$

Which:
$\sum_{i=0}^{S^2}$ is for each grid cell
$\sum_{i=0}^{S^2}$ is for each box
1_{ij}^{obj} is one if the object appears in cell i and j-th box detect it, otherwise 0
C_i is the ground truth confidence score
$\widehat{C_i}$ is predicted confidence score
$\widehat{p_i}(c)$ denotes the conditional class probability for class c in cell i
λ_{coord} increase the weight for the loss in the boundary box coordinates
λ_{noorbj} set to 0.5 to decrease the loss for empty boxes
1_{ij}^{noobj} is one if there is no object in the i-th cell, otherwise 0

2.6 Evaluation of Deep learning's Model

A confusion matrix is a technique to summarize the performance of the model being trained used to detect the status of the eyes of the driver. The number of correct and incorrect predictions are summarized with count value and break down by each class. The confusion matrix output shows the errors being made by the model, but most importantly the types of error being made by the model to detect. In this application

Table 1 Confusion matrix

	Open eyes	Close eyes
Open eyes	True positive	False positive
Close eyes	True negative	False negative

to detect the status of the eyes of the driver, the model is trained to detect two classes, which are open eyes and closed eyes. As a result, two-row by two-column confusion matrices will be applied to this application. The event row will be assigned to "positive" while the no event row will be assigned to "negative". Meanwhile, the event column of the prediction is "true" and no event is "false". In this application to detect the status of the eyes of the driver, "True positive" is assigned to predict open eyes correctly, "False positive" is assigned to mispredict open eyes. In contrast, "True negative" is assigned to predict close eyes correctly while "False Negative" is assigned to predict close eyes wrongly, as shown in Table 1.

With these parameters from the confusion matrix, the accuracy, precision, recall and Fl score can be determined. Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted, which included correctly predict open eyes and correctly predicted close eyes over the total test dataset, and the formula is as Eq. 2. Precision is the ratio of the correctly predicted open eyes over the total number of open eyes and represented in Eq. 3. Recall or called sensitivity, is the ratio of correctly predicted open eyes to the correct total test dataset, and the mathematical representation shown in Eq. 4. F1 score is the weighted average of Precision and Recall. As a result, the F1 score considers both false positive and false negative. Fl score represents the harmonic score evaluation of precision and sensitivity. It represents an unbiased accuracy since it does not affect by imbalanced data, the mathematical representation shown in Eqs. 2 to 5.

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Prediction} \tag{2}$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{3}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{4}$$

$$F1score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity} \tag{5}$$

3 Results and Discussion

Performance of the developed model is being evaluated by using accuracy, precision, recall and Fl score. An improvement version of the confusion matrix was used, which is the two-by-three matrix used instead of the two-by-two matric due to some images that cannot be detected. As a result, the third column is added to the confusion matrix for the images that cannot be detected to provide more apparent evaluation results.

The tested data which are used to evaluate the performance of the model to detect the status of the eyes of the driver consists of ten percent of the dataset, which is not included in training model. The evaluated model's performance using Fl score below 60% was set as threshold of not applicable to real life. After that, the combined dataset had been processed by filtering out the defective data and the model is retrained by using new dataset. The deep learning model to detect the status of the driver's eyes reached 86.58% accuracy in Fl score the processed dataset. Transfer learning approach was implemented in this application to transfer the weightage of the neural network that has learned from previous tasks and apply that weightage to detect the status of the driver's eyes.

3.1 Eye Closure Detection

The trained model to detect the status of the eyes of the driver was tested with a personal computer and a webcam. The trained model is applied with the PERCLOS method to measure the drowsiness level of the driver. As mentioned previously, with the average frame rate of 30 frames per second, 450 frames will be captured and will take into consideration for PERCLOS, if the PERCLOS value is larger than 20%, the state of the driver will become "sleepy". Figure 5(a) shows the status of the eyes of the driver which is detected by using the trained model when the status of the eyes of the driver is in open eyes state. From the figure, the driver is in open eyes state and the model is also able to detect that the status of the eyes of the driver is in open eyes state. Figure 5(b) shows the status of the eyes of the driver which is detected

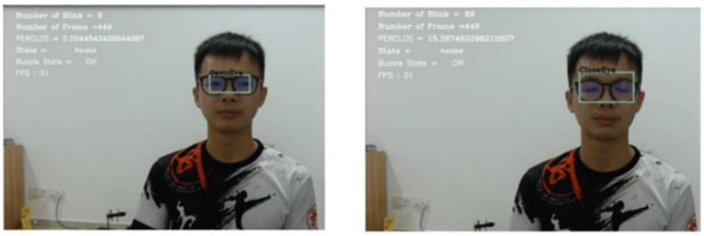
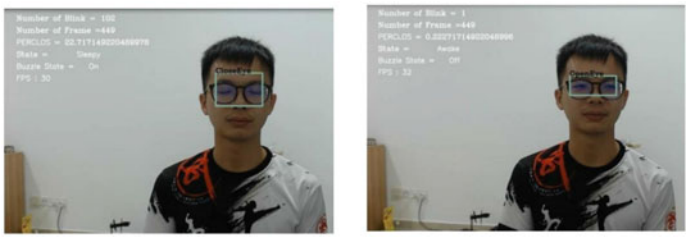


Fig. 5 **a** Status of the eyes of the driver when in open eyes and **b** Status of the eyes of the driver when in closed eyes



```
Number of Blink = 102
Number of Frame =449
PERCLOS = 22.717149220489978
State =        Sleepy
Buzzle State =    On
FPS : 30
```

```
Number of Blink = 1
Number of Frame =449
PERCLOS = 0.22271714922048996
State =        Awake
Buzzle State =    Off
FPS : 32
```

(a) (b)

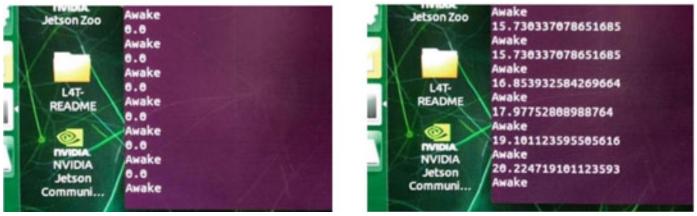
Fig. 6 **a** Drowsiness level of the driver when the PERCLOS is higher than 20% and **b** Drowsiness level of the driver when the PERCLOS is lower than 20%

by using the trained model when the status of the eyes of the driver is in close eyes state. From the figure, the driver is in open eyes state and the model is also able to detect that the status of the eyes of the driver is in open eyes state. Figure 6(a) shows the drowsiness state of the driver when the PERCLOS value is higher than 20%, the state of the driver is changed to drowsy. Figure 6(b) shows the drowsiness state of the driver when the PERCLOS value is lower than 20%, the state of the driver is changed to awake.

3.2 Deep Learning Model Tests with PERCLOS on NVIDIA Jetson Nano

A monitor display was connected to the NVIDIA Jetson Nano, to view the PERCLOS value and the buzzer state in a terminal. In this development, the state of the driver will be sleepy when the PERCLOS value is higher than 20%, explaining when the state of the driver will become sleepy if his eyes are closed more than 20% over a period of time.

Figure 7(a) is showing that the deep learning model is detecting that the driver is in open eyes state, the PERCLOS value does not add as there is no closing eye.

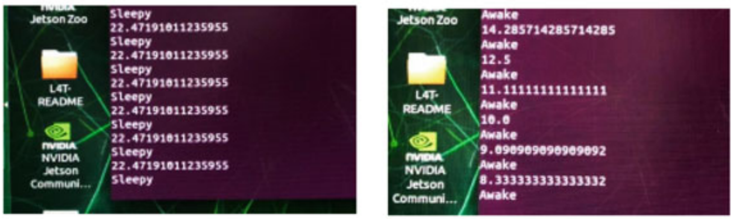


```
Awake
0.0
Awake
0.0
Awake
0.0
Awake
0.0
Awake
0.0
Awake
0.0
Awake
```

```
Awake
15.730337078651685
Awake
15.730337078651685
Awake
16.853932584269664
Awake
17.97752808988764
Awake
19.101123595505616
Awake
20.224719101123593
Awake
```

(a) (b)

Fig. 7 **a** The PERCLOS value is not adding when the driver is on open eyes state and **b** The PERLOS value is adding when the driver is in close eyes state



```
Sleepy
22.47191011235955
Sleepy
22.47191011235955
Sleepy
22.47191011235955
Sleepy
22.47191011235955
Sleepy
22.47191011235955
Sleepy
22.47191011235955
Sleepy
```

(a)

```
Awake
14.285714285714285
Awake
12.5
Awake
11.11111111111111
Awake
10.0
Awake
9.090909090909092
Awake
8.333333333333332
Awake
```

(b)

Fig. 8 **a** The state of the driver is changed to sleepy when the PERCLOS is higher than 20% and **b** The state of the driver is changing back to awake when the PERCLOS is lower than 20%

Figure 7(b) is when the driver is closing his eyes, the figure shows that the PERCLOS value is adding as the number of close eyes is adding over the specific amount of time. However, the state of the driver is awake as the PERCLOS value is below 20%.

When the PERCLOS value is higher than 20%, the state of the driver changed to sleepy. It is shown in Fig. 8(a) when the PERCLOS value is higher than 20%, the state of the driver is becoming sleepy. While the PERCLOS value is dropping to below 20%, the state of the driver will become awake as shown in Fig. 8(b). The buzzer will be triggered as well when the state of the driver is becoming sleepy.

4 Conclusion

This paper reported algorithm development and testing to provides a solution to the problem of detecting the state of drowsiness using arithmetic based method. This system uses percentage of eye closure method to detect sleepiness. Deep learning method was chosen and carried out to the combined datasets of from CEW dataset and KomNet. The algorithm development was based on the concept of eye-tracking. To obtain finer results, three thousand six hundred and thirty-two images from these datasets have been used. The developed algorithm was partially tested and found to be effective and able to provide response to the system from the buzzer. This developed method provides a non-intrusive approach for drowsiness detection. In future research, a few areas of improvement were suggested such as the frequency of yawning can also be used as a feature parameter to detect drowsiness and self-training mode for face recognition and eyes dataset of the driver was recommended to personalize the development application and minimizing inaccurate output.

References

1. Rajkumarsingh B, Totah D (2021) Drowsiness detection using android application and mobile vision face API. R & D J South Afr Inst Mech Eng 37(1):26–34

2. Zhipeng M, Shuwan Y, Jian Z, Jingjiu Q, Jun S, Jingshuang D (2018) Research on drowsy-driving monitoring and warning system based on multi-feature comprehensive evaluation. IFAC Papers Line 5131(1):784–789
3. Karthikeyan S, Subha R, Elakkiya G, Kowshika R, Dhanvarshni S (2021) Driver-drowsiness detection system. Turk J Comput Math Educ 12(9):1892–1898
4. Rukhsar K, Shruti M, Shivraj P, Suraj A, Saritha LR (2019) Human drowsiness detection system. Int J Eng Adv Technol (IJEAT) 8(4):2249–8958
5. Jabbar R, Shinoy M, Kharbeche M, Al-Khalifa K, Krichen M, Barkaoui K (2020) Driver drowsiness detection model using convolutional neural networks techniques for android application. In: 2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT), vol 2, no 1, pp 237–242
6. Sahayadhas A, Sundaraj K, Murugappan M (2012) Detecting driver drowsiness based on sensors: a review. Sensors (Basel) 12(12):16937–16953
7. Prakash C, Rahul S, Gautam S, Smarjeet DA (2016) Survey paper on drowsiness detection & alarm system for drivers. Int Res J Eng Technol (IRJET) 3(12):1433–1437
8. Messaoud D, Abdelmadjid B, Veronique C (2018) A light on physiological sensors for efficient driver drowsiness detection system. Sensors Transd J 224(8):39–50
9. Kiran VBN, Raksha R, Anisoor R, Varsha KN, Nagamani NP (2020) Driver drowsiness detection. Int J Eng Res Technol (IJERT) 8(15):33–35
10. Maryam H, Alireza M, Aliasghar BS (2020) Driver safety development: real-time driver drowsiness detection system based on convolutional neural network. SN Comput Sci 1(289):1–10
11. Mohsen P, Adel M, Gebraeil NS, Mohammad MB, Alireza K, Mohammad HE (2018) Using image processing in the proposed drowsiness detection system design. Iran J Public Health 47(9):1371–1378
12. Walid M, Belhassen A, Roobaea A, Abdulmajeed A (2019) Automated drowsiness detection through facial features analysis. Computación y Sistemas 23(2):511–521
13. Redmon AJ, Farhadi A (2016) YOLO: real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, no 2, pp 779–788
14. Redmon J, Farhadi A (2017) YOL09000: Better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1, no 2, pp 1–9
15. Zhao ZQ, Zheng P, Xu ST, Wu X (2019) Object detection with deep leaming: a review. IEEE Trans Neural Netw Learn Syst 1(2):3212–3232
16. Feiping N, Zhanxuan H, Xuelong L (2018) An investigation for loss functions widely used in machine learning. Commun Inf Syst 18(1):37–52