# DrowsyDetect: Android Application for Driver Drowsiness Detection Using Eye Closure Rate with Deep Learning

Furwa Asim[1] and Smitha Sunil[1#]

[1]Middle East College, Muscat, Oman
[#]Advisor

## ABSTRACT

Studies on sleep-related crashes show that about 15% of accidents are classified as sleep-related accidents. Furthermore, daytime sleepiness is common in young drivers. This means the chances of car crashes due to drowsiness are high. The proposed research attempts to tackle this issue and alert the driver in case of drowsiness and bring the driver back to the consciousness of the surroundings to avoid crashes. The current study aims to develop an Android mobile application, which will be used for drowsiness detection in drivers. It compares multiple Deep Learning, Convolutional Neural Network (CNN) models including MobileNet, VGG16, and a custom CNN to detect user's faces and analyze the eye closure rate. The models use Haar Cascade to detect the human face and eyes. If the closure rate goes above a certain threshold, an alarm will be triggered on the mobile phone alerting the driver of their drowsiness. The trigger will then be logged on to the user account to allow users to access the logs later and analyze their drowsiness patterns also. The proposed system is simplified to eliminate the need for external equipment like head mounts or sensors that may cause driver discomfort. However, there is scope for enhancing the app by adding complex features such as sleep schedules, sleep-related information, facts, emergency triggers, and so on.

## Introduction

Humans have busy routines in the modern day due to work and responsibilities. This makes it difficult to manage time, resulting in a lack of sleep. This may cause sleep-related drowsiness for drivers, causing a lack of focus on the road and lower alertness to react on time. Becoming drowsy whilst driving can serve as a huge risk not only to the driver's life but also to passengers. By losing focus, the driver becomes prone to vehicle accidents. Vehicles generally travel between the speed of 100-120 km/h on the main roads. At these speeds, the time allowed to the driver between detecting danger and reacting to the detected danger is very low. Adding drowsiness to the already low reaction time on this high of a speed is riskier than that on lower speeds. High-speed crashes can lead to severe injuries or worse, loss of life.

The proposed research develops an application that can alert the driver once they are detected to be drowsy through an alarm that is triggered on their mobile phone. It helps the driver gain consciousness back and take preventative measures such as resting on the side or taking a break to freshen up before they continue driving. It is expected that using this technology will decrease the number of sleep-related accidents. The application also provides the user with a summary of their drowsiness. This summary will let the user analyze their sleep patterns in comparison to drowsiness and help them find an optimal routine where they have sufficient sleep before driving or investigate issues other than sleep that might make them drowsy. Furthermore, the research is based on deep learning, hence the training is automated without the need for human intervention. This allows to achieve better accuracy in a shorter duration of time to detect drowsiness and send an alert. The paper is structured in five parts, including related works, design, implementation, results, and conclusion.

## Related Works

Singh and Papanikolopoulos (1999) suggested a vison-based, non-invasive approach to determine the drowsiness level in a driver. They use a color video camera, pointing directly at the driver's face. The video input from this camera is used to detect microsleep patterns by tracking the driver's eyes. If the eyes are closed for 5-6 seconds, the person is detected to be drowsy. It is claimed that the accuracy of the system is 95% with up to 30-degree head tilt and 45-degree head rotation tolerance. The drawback of this system, however, is in a practical driving scenario, the head movement will reduce the accuracy of the program, resulting in more frequent false alarms. In a study by Sayed and Eskandarian (2001), they used Artificial Neural Networks (ANN) to detect driver drowsiness. This detection is based on the driving pattern of the driver. The ANN uses the steering angles of the vehicle to detect any sudden changes in the angle. The data was collected using a driving simulator. The results had an accuracy of 90%. However, an issue with this method of drowsiness detection is that it cannot alert the driver before the car is out of control, which leaves the driver with a lower time to respond.

There have also been attempts to detect drowsiness using intrusive methods that use sensors attached to the driver. An example of this is the method proposed by Zhang, Wang, and Fu (2014). They measure the fatigue level of the driver by using signals from an electroencephalogram, electromyogram, and electrooculogram. This data is fed to the ANN to determine the fatigue level of the driver. Although the accuracy of this proposition is extremely high (96.5% - 99.5%), the major issue with it is that it is an intrusive method which means the user will not only have to purchase additional equipment, but also must wear sensors that might affect the quality of driving.

In the research by Lashkov, Kashevnik, Shilov, Parfenov and Shabaev (2019), they use built-in camera phone camera to track the facial features of the driver through Android devices. These features include head movements, mouth state, and eye state. The features are used to analyze the fatigue level of the driver as well as their focus level. An early warning is given in case the driver is drowsy or has lost focus. It uses CNN with OpenCV and Dlib with the COCO dataset for this purpose and has both online and offline modes. One of the issues with this, however, is that the application needs to track multiple features of the face, hence requiring heavy processing. This, in turn, means that a higher-end Android device will be required to work which might not be accessible by everyone.

Rahman, Islam et al (2020) propose a Convolutional Neural Network (CNN) called EyeNet as shown in Figure 1. This model is used for eye state classification tested over three datasets. These include CEW, ZJU, and MRL Eye. The model is evaluated under different conditions such as lighting, reflection, devices, etc. The accuracy is shown to be 99%, which is a 3% improvement from classic previous eye states classifiers such as SVMs or shallow neural networks. To create CNN, the authors used Keras with a TensorFlow backend. The highest accuracy is shown when the model is trained through the ZJU dataset and tested against the CEW dataset.

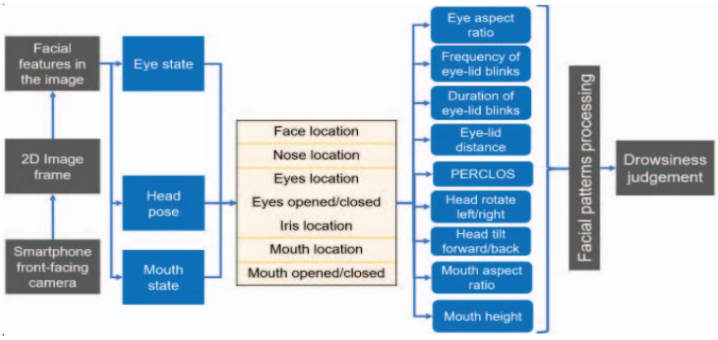


**Figure 1.** General scheme for drowsiness state detection (Lashkov et. al., 2019)

## Design

The general design of the system is illustrated in Figure 2. The video input from the mobile phone's camera is used to detect eyes on the user face. This is then run through the trained model to classify the eyes as opened or closed. If the eyes remain closed for a certain time, the alarm on the phone is triggered, and the event is logged. The system pipeline is shown in Figure 3. First, the data is gathered, next it is preprocessed. Since the closed eyes and open eyes class data contains 2000 images each, the dataset is balanced, hence does not require to be balanced. After which it is split into training and validation. After this, the model is built and trained, followed by an evaluation, fine tuning (if needed), and finally deploying the project.

## Implementation

### Dataset

The dataset (Patil, 2021) used for training the DL model is based on the MRL dataset. The MRL dataset (MRL, 2021) consists of data from 37 people, where 8 properties for each person are captured including gender, glasses, eye state, etc. From this dataset, the used dataset takes the images related to the eye state, and the data is split into two folders: one for open eyes, and the other for closed eyes. These folders contain mixed images from the participants, 2000 images in each folder. This makes the total number of images 4000. The images are in Portable Network Graphics (PNG) format, approximately 4KB per image. This dataset was chosen as it is based on a popular dataset, and has a variety of eye images, hence allowing the model to be trained with higher accuracy. The MRL dataset is reference by multiple literatures, including the research by Rahman, et al (2020), as discussed in the literature review section.

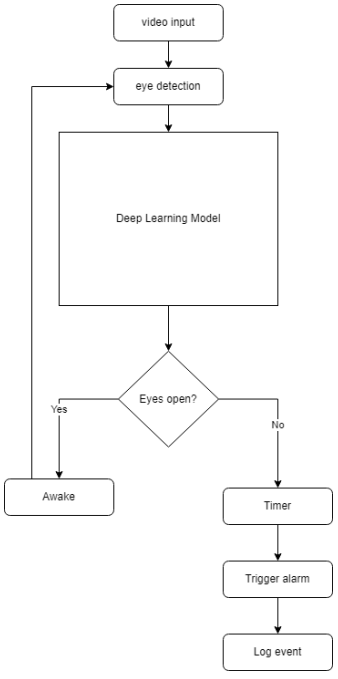


**Figure 2.** General System Design

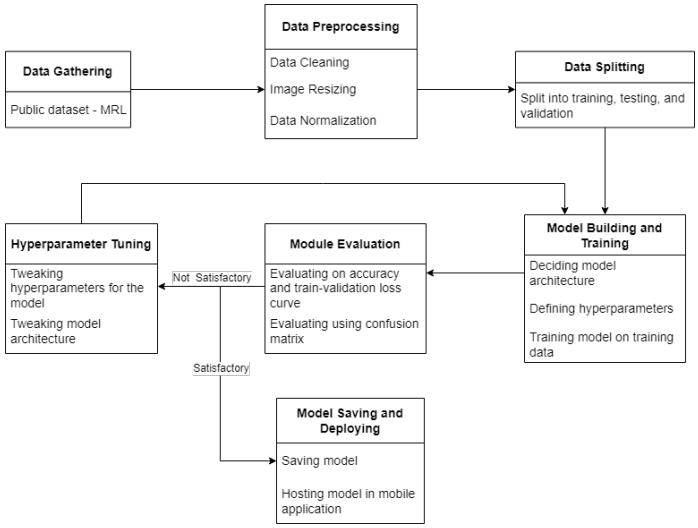


**Figure 3.** Pipeline Flow

## Preprocessing

The dataset is evenly split as two-thousand images for each class (opened eyes or closed eyes), hence the need for balancing the dataset is eliminated. The images are resized to 244 by 244 pixels to give input to the models for training. The data is normalized and serialized using the pickle library.

## Architecture

All models are trained on 15 epochs with an *earlystopping* callback with a patience of 3 to avoid overfitting. The learning rate is set to be 0.1, with a binary cross entropy loss, Adam optimizer, and Sigmoid activation function. All models also use the same data pre-processing techniques for the same purpose. The models use feature extraction using Haar Cascade to detect the human face and eyes, which is then used for classification. The hyperparameters for all models are kept constant for a better comparison.

## Training Device

The device used to train the model is an HP Omen 15. It has a 64-bit Windows 10 operating system, Intel Core i7 6 core processor, Nvidia GeForce GTX 1070 Max-Q GPU, 6 GB CPU, 32 GB RAM, and 256 GB SSD along with 1 TB HDD.

### *MobileNet Architecture*

This CNN architecture is one of the architectures provided by Keras. It uses depth-wise separable convolutions for building lightweight, but deep, neural networks. The model uses transfer learning from the pre-trained MobilNet Architecture. The input form the first layer, and the output from the fourth layer from the bottom up is taken as the base input and output. Next, the flatten, dense, and activation layers are added, which use the base output. This decreases the trainable parameters by approximately one million. The model is then compiled and trained for evaluation.

### *VGG16 Architecture*

The VGG16 CNN architecture uses the pre-trained VGG16 architecture form Keras, consisting of 16 CNN layers. The weights used for this model are the default weights used to train the ImageNet dataset, since these weights seem to be the most optimal for this architecture. The first layer is used as the base input, and the second last layer is used as the base output. A flattened, dense, and activation layer is then added, after which the model is compiled and trained. The reduction in the trainable parameters is one twenty-four million. Next, the model is compiled, trained, and evaluated.

### *Custom Architecture*

This CNN architecture is based on 9 custom layers added to the model. Mainly consisting of convolutional and batch normalization layers, the architecture can be shown in Figure 4.

```
Layer (type)                 Output Shape
=================================================
conv2d (Conv2D)              (None, 222, 222, 32)

max_pooling2d (MaxPooling2D  (None, 111, 111, 32)
)

batch_normalization (BatchN  (None, 111, 111, 32)
ormalization)

conv2d_1 (Conv2D)            (None, 109, 109, 32)

max_pooling2d_1 (MaxPooling  (None, 54, 54, 32)
2D)

batch_normalization_1 (Batc  (None, 54, 54, 32)
hNormalization)

conv2d_2 (Conv2D)            (None, 52, 52, 64)

max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)
2D)

batch_normalization_2 (Batc  (None, 26, 26, 64)
hNormalization)

conv2d_3 (Conv2D)            (None, 24, 24, 64)

max_pooling2d_3 (MaxPooling  (None, 12, 12, 64)
2D)

batch_normalization_3 (Batc  (None, 12, 12, 64)
hNormalization)

flatten (Flatten)            (None, 9216)

dense (Dense)                (None, 1)
```

**Figure 4.** Custom CNN Architecture

Once the architecture is developed, the model is compiled and trained based on the architecture.

## Results

Using the MobileNet architecture, both the training accuracy and the validation accuracy is 100%. It uses 5/15 epochs to achieve this due to early stopping. The Training-Validation Loss Curve is depicted in Figure 5.

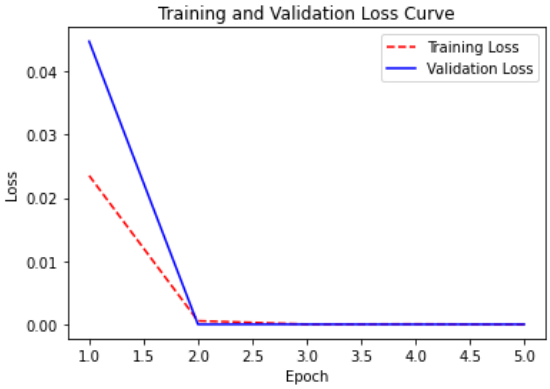


**Figure 5.** Training-Validation Loss Curve – MobileNet

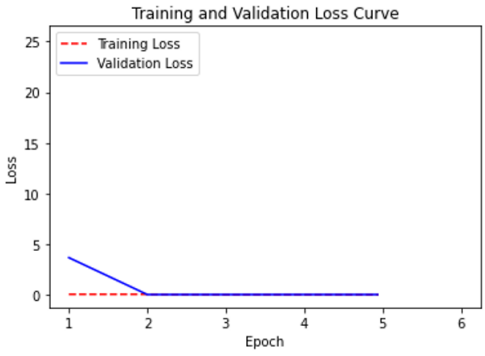


**Figure 6.** Training-Validation Loss Curve – VGG16

The model that uses the VGG16 architecture also presents an accuracy of 100%, however, it takes 6/15 epochs to train the model, with early stopping. Figure 6 shows the Training and Validation Loss Curve.

The custom architecture gives an accuracy of 98.75% and takes 10/15 epochs to train the model based on earlystopping callback. The Training-Validation Loss Curve is illustrated in Figure 1.

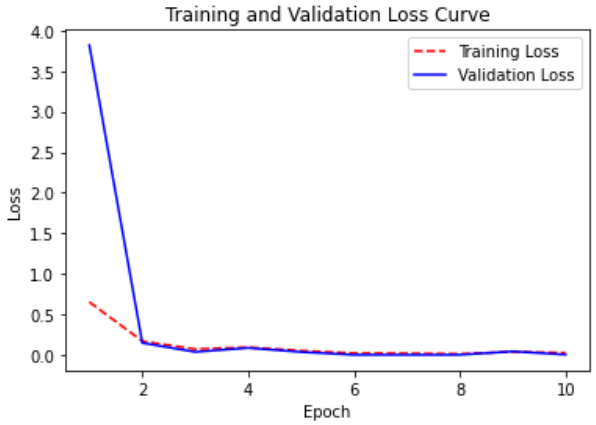


**Figure 1.** Training-Validation Loss Curve – Custom Architecture

The accuracy obtained for each model is presented in Table 1.

**Table 1.** Accuracy Comparison

| **Model** | **Accuracy (%)** |
|---|---|
| MobileNet | 100 |
| VGG16 | 100 |
| Custom | 98.75 |

## Conclusion

It is concluded that both MobileNet and VGG16 have the best accuracy out of the three models considered, however, since MobileNet is less intensive on the computational power, this model will be used for the mobile application to allow a wider range of devices to handle the model. In the future, the system can include additional features such as sending notifications to emergency contacts. Furthermore, the accuracy of the model can be improved by incorporating other facial features such as yawning, head tilt, etc. In addition, the model can also be trained to provide better results in different environments such as low-lit areas.

## References

Al-Abri, M. A., Al-Adawi, S. A., Al-Abri, I., Al-Abri, F., Dorvlo, A., Wesonga, R., & Jajou, S. (2018). Daytime Sleepiness Among Young Adult Omani Car Drivers. *SQU Med J, 18*(2), 143-148. 10.18295/squmj.2018.18.02.004

Filtness A.J., Armstrong K.A., Watson A. & Smith S.S. (2017). Sleep-related crash characteristics: Implications for applying a fatigue definition to crash reports. *Accident Analysis & Prevention*, *99*(B), 440-444. https://doi.org/10.1016/j.aap.2015.11.024

Lashkov, I., Kashevnik, A., Shilov, N., Parfenov, V. & Shabaev, A. (2019). Driver Dangerous State Detection Based on OpenCV & Dlib Libraries Using Mobile Video Processing. *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. 74-79. doi: 10.1109/CSE/EUC.2019.00024.

MRL. (2021), *MRL Eye Dataset.* http://mrl.cs.vsb.cz/eyedataset

Patil, P. V. (2021). *Drowsiness Detection Dataset.* https://www.kaggle.com/datasets/prasadvpatil/mrl-dataset

Rahman, M. M., Islam, M. S., Ara Jannat, M. K., Rahman, M. H., Arifuzzaman, M., Sassi, R., & Aktaruzzaman, M. (2020). EyeNet: An Improved Eye States Classification System using Convolutional Neural Network. *International Conference on Advanced Communication Technology (ICACT)*, 84-90. https://doi-org.masader.idm.oclc.org/10.23919/ICACT48636.2020.9061472

Sayed, R. & Eskandarian, A. (2001). Unobtrusive drowsiness detection by neural network learning of driver steering. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *215*(9), 969-975. doi:10.1243/0954407011528536

Singh, S. & Papanikolopoulos, N. P. (1999). Monitoring driver fatigue using facial analysis techniques. *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, 314-318. doi: 10.1109/ITSC.1999.821073

Zhang, C., Wang, H., & Fu, R. (2014, February). Automated Detection of Driver Fatigue Based on Entropy and Complexity Measures. *IEEE Transactions on Intelligent Transportation Systems*, *15*(1), 168-177. doi: 10.1109/TITS.2013.2275192.