Driver Monitoring using Deep Learning

Younsuk Choi
Department of Electronic Engineering
Hochschule Hamm Lippstadt
Lippstadt, Germany
younsuk.choi@stud.hshl.de

***Abstract*—The paper presents a driver monitoring system utilizing deep learning, focusing on the detection of drowsiness through facial and posture analysis. The motivation stems from the alarming global statistics on road traffic casualties, with 95 percent of fatal accidents attributed to driver errors. The proposed system leverages Convolutional Neural Networks (CNNs), specifically the EfficientNetB0 architecture, to analyze real-time video streams captured by a camera facing the driver. Key aspects include the behavioral features for drowsiness detection and the integration of IoT technology for timely alerts. The implementation involves the MRL Eye Dataset for training, and the results showcase the model's efficiency with a focus on accuracy, model structure, and performance metrics. The paper provides insights into CNN architecture, transfer learning, and the potential impact of the proposed system on road safety.**

I. Introduction

According to the [1], approximately 1,19 million road traffic casualties were reported globally in 2021. This number indicates that 15 per 100,000 human lives were taken by traffic accidents. [1] Especially, according to their 2019 data, road traffic injury is one of the leading causes of mortality for people aged 5 to 29 years old. [1] It is truly unfortunate that man-made errors like road traffic casualty are the leading cause of death for people of this age range, even greater than any kind of illness. Furthermore, the report estimates that the global macroeconomic cost of the traffic accident amounts to 1,8 trillion dollars, which is as huge as 10 to 12 percent of the global gross domestic product. [1] This poses an important question to the health and development challenge to our global community.

There are a variety of factors that contribute to the road traffic accident. Those are driver-related issues, technical failures, and environmental conditions such as weather, road surface conditions, or obstacles on the road. For driver-related issues, problems such as exceeding the speed limit, drunk driving, non-use of seatbelts, distracted driving, etc are contributing to traffic accidents. [1] What is noteworthy is that most of fatal traffic accidents arise from driver errors. The driver errors in this context have three main factors: alcohol abuse, drowsiness, and inattention. [2] In the context of driver drowsiness, this is what could be easily prevented with the help of recent technology, the driver alerting system with the early detection of drowsiness.

Drowsiness detection encompasses four primary aspects: vehicle-based, subjective, physiological, and behavioral. [2] Vehicle-based measures involve monitoring steering wheel status, pedal input, car acceleration, and speed. [2] Subjective measures rely on self-assessment of sleepiness levels, commonly gauged using the Stanford Sleepiness Scale (SSS) and Karolinska Sleepiness Scale (KSS). [2] Physiological measurement employs electrocardiography (ECG) and electroencephalography (EEG) for the measurement process. [2] Among various methods, accurate measurement of driver drowsiness can be achieved with PERCLOS (Percentage of Eye Closure) measurement standing out prominently. [2]

Lastly, and most importantly, the behavioral measures evaluate various actions, including monitoring eye blinks over a specific time interval, assessing head pose, steering wheel gripping pressure, facial detection, yawning detection, and eye gaze. [2] Eye gaze measurement involves the utilization of eye corners and iris centers, employing a linear mapping function to ascertain the direction of the eyes. [2] Pupil center detection relies on a deformable eye based on shape and intensity, utilizing movement decision algorithms, with sensitivity to accuracy and diminished performance noted in low-resolution video sequences. [2] The development of the POSIT algorithm (Pose from Orthography and Scaling with Iterations) is facilitated by techniques in head pose estimation and eye gaze. [2]

For the driver alerting system of drowsiness, the behavioral aspect can be considered most suitable because the evaluation of drowsiness from the driver's actions can be achieved with the help of deep learning technology. In this paper, we restrict our discussion to the behavioral aspect of drowsiness detection, furthermore, detection of drowsiness based on imagery inputs for driver's posture and facial features. The remainder of the paper is structured as follows. In Section II we present a brief overview of CNNs. Our proposed solution is discussed in Section III. Section IV explains how the implementations are performed, to what extent this implementation is conducted, and what the results are. Section V concludes our paper based on what we discussed.

II. Preliminaries information on Convolutional Neural Network

A. Architecture of CNN

A Convolutional Neural Network (CNN) is a type of deep learning architecture designed for processing and analyzing visual data, particularly images. [3] CNNs are specialized in dealing with image data such as image classification. CNN is a powerful tool as it is spatially independent, meaning that it

has the power for feature extraction. [4] Also when an imagery input goes through deeper layers, the power of gaining abstract features is augmented, thus being able to detect higher-level features. [4] CNNs are constructed with some fundamental layers such as convolutional layers, pooling layers, and fully connected layers. [3] As seen in Figure 1, a standard CNN has an architecture of repeated sets of convolutional layers and a subsequent pooling layer, along with fully connected layers. [3]

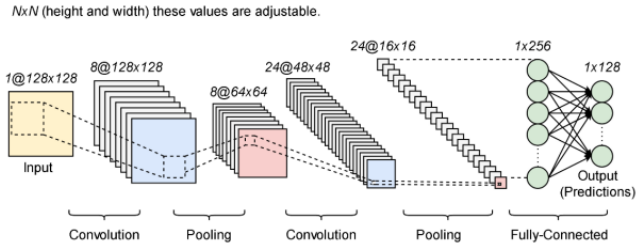

Fig. 1. Example of CNN Architecture
[5]

When an image is fed into CNN as an input, it recognizes the image as an array of numbers. [3] On the convolutional layer, this input array is multiplied with a sliding kernel, which is a smaller-sized array, to create a feature map, which stores the characteristics of the input array while reducing its size. [3] This operation is called convolution and it can be repeated several times both in parallel for more feature maps and in a hierarchical manner for feature extraction. [3] Then, with the activation function, which is a set of mathematical operations conducted to determine the activation of neurons on the layer. [3] Next, on the pooling layer, the feature maps are reduced in size and complexity and this downsampling gives spatial hierarchies to different levels of feature maps. [3] This operation adds stability to the network so that it is more resilient to distortion or translation, which eventually augments pattern recognition. [4] These final feature maps on each layer are then flattened and linked to fully connected layers, or dense layers in other words, where neurons are present like traditional neural networks and activated based on activation function. [4] Each node in this dense layer is directly linked to every node in the previous and subsequent layers, making a system of matrix. [4] In this way, the matrix over these fully connected layers can make predictions from the patterns and relationships in the extracted features that are captured by previous layers. [4]

When training the CNN, loss function also takes an important role. [3] The choice of a loss function depends on the task. For image classification, categorical cross-entropy is often used. It measures the dissimilarity between predicted and true class probabilities. [3] Stochastic Gradient Descent (SGD) or advanced optimizers like Adam are employed to minimize the loss function. [3] These optimization algorithms adjust the network's parameters (weights and biases) to reduce prediction errors. [3] Backpropagation involves adjusting the weights by considering the gradient of the loss to the parameters of the network. [3] It involves propagating the error backward through the network and adjusting weights using the chain rule of calculus.

To recapitulate, Convolutional Neural Networks are designed to learn hierarchical representations of visual data. [3] The convolutional and pooling layers are for local feature recognition, while fully connected layers provide some sense of matrix for global feature and prediction. [3] Furthermore, activation functions share non-linear characteristics, and backpropagation enables learning of the most suitable kernel arrangement. [3] With advancements like data augmentation, dropout, and regularization techniques, CNNs have become powerful tools in image analysis, driving breakthroughs in computer vision applications. [3]

B. Transfer Learning

Transfer Learning is an effective strategy that is used widely in order to overcome limited datasets. [3] A deep learning model can be pre-trained on a vast dataset and used for other models, especially for the convolutional layers of CNN that are designed for another task. The core advantage of this method is that features extracted from other datasets can be employed for tasks with small datasets, which helps save time for training or enhance performance because smaller datasets lack in the amount of data, thus being unable to reach high accuracy. [3] There are many pre-trained models and this paper focuses on EfficientNetB0 due to its relevance to our implementation.

III. Proposed Work

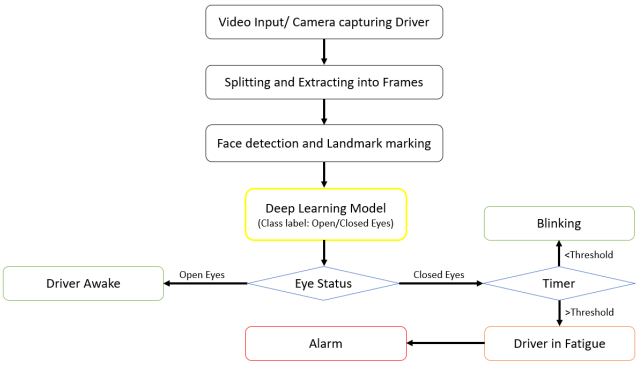

Fig. 2. Block Diagram of Proposed System

The proposed driver alerting system for drowsiness has the following block diagram for the system description as shown in Figure 2. The system incorporates both IoT technology and DL algorithms to detect the driver's drowsiness and alert the user with an alarm. The main hardware and components that are to be used for the systems are a microcontroller, a camera, a set of tools for alarming, and a deep-learning interface. First of all, a camera is situated in front of the driver to capture the video stream of the driver's upper body features. Then the video stream will be split into frames so that it could be fed into the deep learning network. With the deep learning

algorithm, based on the real-time input image extracted from the stream, it will return the eye status of the driver, in other words, whether they are opened or closed. This output will be used in the microcontroller to tell the driver's status with the timer whether the driver is awake, blinking, or dozing, based on the length of time the driver's eyes are regarded closed. Once it detects the driver dozing, the alarm will be triggered to wake the driver up. This system proposal was designed 100 percent by the author of this paper, after studying through [8], [9], [10], [11].

IV. Implementation and Results

A. Dataset Information

For the implementation, the MRL Eye Dataset, released on Kaggle in 2021, was used. The dataset contains 84,898 images, having a size of 224 × 224 pixels with 3 channels, divided into train and test data where images are labeled to be eye-opened and eye-closed. [6]

B. Implementation and Architecture

The Convolutional Neural Network (CNN) architecture implemented in the provided code is built upon the EfficientNetB0 architecture, renowned for its efficiency in balancing model capacity and computational resources. EfficientNetB0 is well-suited for image classification tasks, offering high accuracy with fewer parameters compared to traditional architectures. The implementation is conducted on rather a shallow level to see if the trained model can tell whether an input image is an image of the closed eye or the opened eye. The code implementation was conducted after studying through [7]

C. Results

1) Images of the Eyes and Labeling: The images of eyes, a critical component of the dataset, undergo preprocessing and are fed into the model for training and evaluation. These images are typically of dimensions 224x224 pixels with three color channels. During training, the model learns to differentiate between open and closed eyes based on the distinctive patterns and features extracted through convolutional and pooling layers. The labeling of these images involves assigning them to respective classes (open or closed eyes), forming the basis for model training and evaluation. The visual representation of eyes and their corresponding labels is integral to understanding the model's ability to discern between different eye states.

2) Model Structure: The model's structure is outlined using the 'model.summary()' function. This concise overview encapsulates the number of parameters, layer types, and shapes, providing insights into the model's efficiency. The summary showcases the streamlined architecture of EfficientNetB0, emphasizing its ability to achieve powerful results with a relatively small number of parameters compared to traditional architectures.

3) Classification Report: The model had to be trained with the dataset. It ran through 3 epochs only due to the low processing power of the available CPU. The accuracy in the last epoch turned out to be superior to previous ones, achieving 99.13 percent.

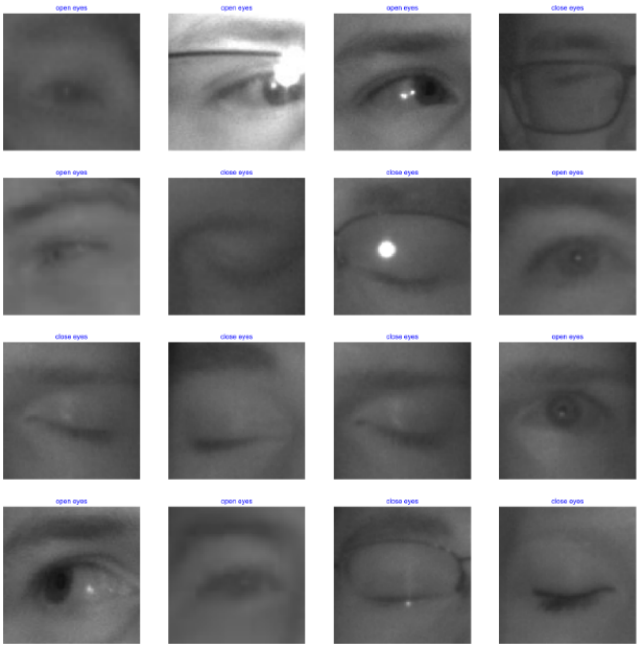

Fig. 3. Images of the Eyes and Labeling

Model: "sequential"

Layer (type)	Output Shape	Param #
efficientnetb0 (Functional)	(None, 1280)	4049571
batch_normalization (BatchNormalization)	(None, 1280)	5120
dense (Dense)	(None, 256)	327936
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

Total params: 4383141 (16.72 MB)
Trainable params: 4338558 (16.55 MB)
Non-trainable params: 44583 (174.16 KB)

Fig. 4. Model Structure

4) Plot Training History: The training history is visualized using matplotlib. The training history plots show the progression of training and validation loss, as well as training and validation accuracy over epochs. These visualizations offer insights into the convergence and performance of the model. The x-axis represents the number of training epochs, while the y-axis reflects the corresponding values of the metrics. The training loss curve demonstrates the decline in training error over epochs, indicating the model's ability to learn from the training data. Similarly, the validation loss curve illustrates the generalization performance on a separate validation set. The training accuracy and validation accuracy curves provide a glimpse into the model's learning behavior, showcasing improvements and potential overfitting.

```
batch_size = 16   # set batch size for training
epochs = 3   # number of all epochs in training

history = model.fit(x= train_gen, epochs= epochs, verbose= 1, validation_data= valid_gen,
                    validation_steps= None, shuffle= False)
```

```
Epoch 1/3
4084/4084 [==============================] - 8527s 2s/step - loss: 0.3333 - accuracy: 0.9840 - val_loss: 0.0931 - val_accuracy:
0.9897
Epoch 2/3
4084/4084 [==============================] - 10013s 2s/step - loss: 0.0946 - accuracy: 0.9897 - val_loss: 0.0743 - val_accurac
y: 0.9901
Epoch 3/3
4084/4084 [==============================] - 7691s 2s/step - loss: 0.0782 - accuracy: 0.9913 - val_loss: 0.0675 - val_accuracy:
0.9909
```

Fig. 5. Classification Report

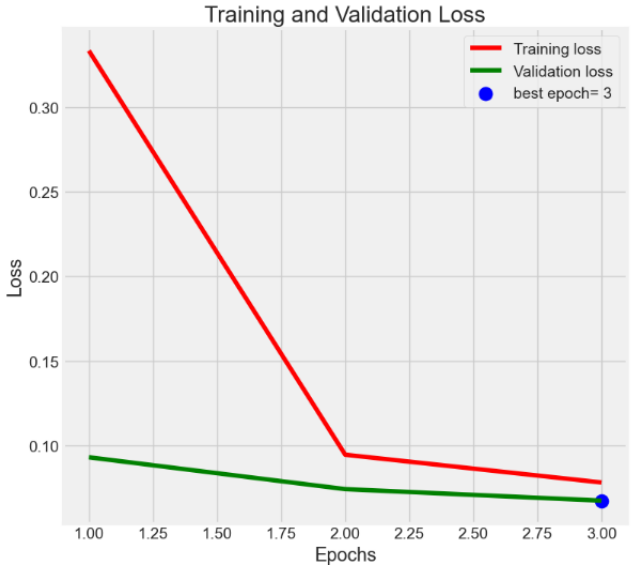

Fig. 6. Plot Training History 1

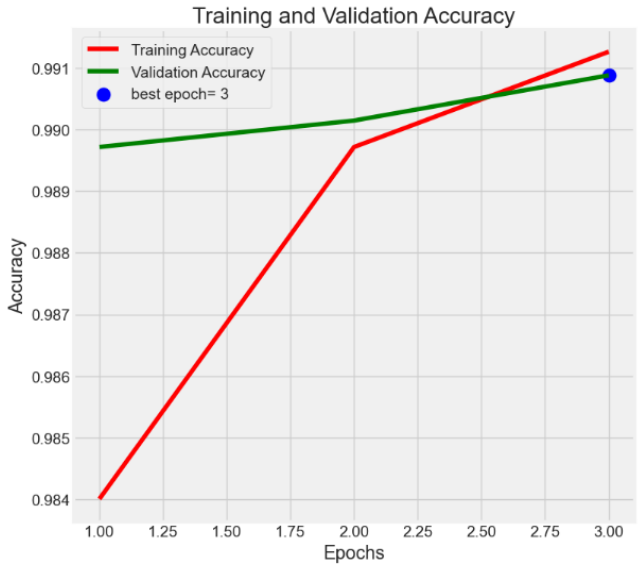

Fig. 7. Plot Training History 2

5) Confusion Matrix: The confusion matrix is a vital tool for evaluating the model's performance in multi-class classification tasks. It visually represents the count of true positive, true negative, false positive, and false negative predictions. Each row of the matrix corresponds to the true class, while each column corresponds to the predicted class. A color scale is often used to highlight the intensity of correct and incorrect predictions. This visual aid helps identify specific classes where the model excels or struggles, providing valuable insights into areas for improvement.

6) Classification Report: The classification report complements the confusion matrix by providing a comprehensive summary of key classification metrics for each class. Precision, recall, and F1-score are commonly included. Precision measures the accuracy of positive predictions, recall gauges the ability to capture all positive instances, and F1 score represents the harmonic mean of precision and recall. These metrics offer a nuanced understanding of the model's performance beyond simple accuracy, especially in scenarios where class imbalances exist.

V. Conclusion

In conclusion, the implemented CNN architecture, based on EfficientNetB0, showcases the integration of key components for effective image classification. Transfer learning, regularization techniques, and visualization tools contribute to the model's efficiency, robustness, and interpretability. Understanding the architectural choices and training procedures, along with insights gained from visualizations like the confusion matrix and classification report, provides valuable context for practitioners and researchers working in the domain of computer vision and deep learning.

References

[1] Global status report on road safety 2023. Geneva: World Health Organization; 2023. Licence: CC BY-NC-SA 3.0 IGO

[2] Hanafi, Mohamad, Nasir, Mohammad, Wani, Sharyar, Abdulghafor, Rawad, Gulzar, Yonis, Hamid, Yasir. (2021). A Real Time Deep Learning Based Driver Monitoring System. International Journal on Perceptive and Cognitive Computing. 7. 79.

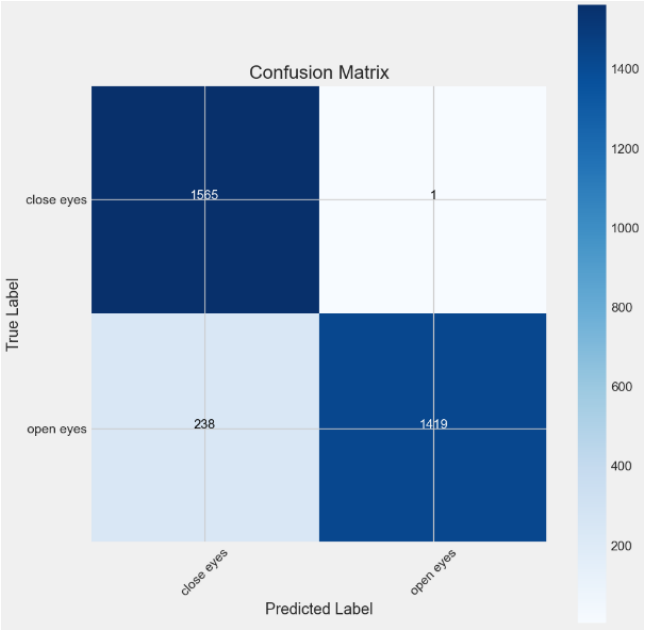

Fig. 8. Confusion Matrix

	precision	recall	f1-score	support
close eyes	0.87	1.00	0.93	1566
open eyes	1.00	0.86	0.92	1657
accuracy			0.93	3223
macro avg	0.93	0.93	0.93	3223
weighted avg	0.94	0.93	0.93	3223

Fig. 9. Classification Report

[3] Yamashita, Rikiya, Nishio, Mizuho, Do, Richard, Togashi, Kaori. (2018). Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 9. 10.1007/s13244-018-0639-9.

[4] Albawi, Saad, Abed Mohammed, Tareq, ALZAWI, Saad. (2017). Understanding of a Convolutional Neural Network. 10.1109/ICEngTechnol.2017.8308186.

[5] Montalbo, Francis Jesmar, Alon, Alvin. (2021). Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears. KSII Transactions on Internet and Information Systems. 15. 147-165. 10.3837/tiis.2021.01.009.

[6] Fusek, R. (2018). Pupil localization using geodesic distance. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 11241 LNCS. 433-444. 10.1007/978-3-030-03801-4,38

[7] https://www.kaggle.com/code/abdallahwagih/human-eyes-detection-open-close/notebookImport-needed-modules

[8] Sood, Naveksha, Sharma, Pranay. (2020). Application of IoT and Machine Learning for Real-time Driver Monitoring and Assisting Device. 10.1109/ICCCNT49239.2020.9225387.

[9] Ch, Venkata, Reddy, U. Srinivasulu, KishoreKolli, Venkata. (2019). Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. Revue d'Intelligence Artificielle. 33. 461-466. 10.18280/ria.330609.

[10] William, P., Shamim, Mohd, Yeruva, Ajay, Gangodkar, Durgaprasad, Vashisht, Swati, Choudhury, Amarendranath. (2022). Deep Learning based Drowsiness Detection and Monitoring using Behavioural Approach. 592-599. 10.1109/ICTACS56270.2022.9987728.

[11] Hashemi, Maryam, Mirrashid, Alireza, Shirazi, Aliasghar. (2020). Deep learning based Driver Distraction and Drowsiness Detection.

VI. Declaration of Originality

I, Younsuk Choi, herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

201223 - Younsuk Choi