Driver Monitoring using Deep Learning

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

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Introduction

According to the [1], approximately 1,19 million road traffic casualty was reported globally in 2021. This number indicates that 15 per 100,000 human lives were taken by traffic accident. Especially, according to their 2019 data, road traffic injury is the one of the leading cause of mortality for people aged 5 to 29 years old. It is truly unfortunate that man-made error like road traffic casualty is the leading cause of death for people of this age range, even greater than any kind of illness. Furthermore, the report estimates that 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. This poses a important question to health and development challenge to our global community.

There are a variety of factors that contributes 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 the driver-related issues, problems such as exceeding speed limit, drink driving, non-use of seat-belts, distracted driving, etc are contributing towards the traffic accident. What is noteworthy is that 95 percent of the fatal traffic accidents arises from driver errors. The driver errors in this context has three main factors: alcohol abuses, 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.

[4] Drowsiness detection encompasses four primary aspects: vehicle-based, subjective, physiological, and behavioral. Vehicle-based involves monitoring steering wheel status, pedal input, car acceleration, and speed [3]. Subjective measures rely on self-assessment of sleepiness levels, commonly gauged using the Stanford Sleepiness Scale (SSS) and Karolinska Sleepiness Scale (KSS) [4]. Physiological measurement employs electrocardiography (ECG) and electroencephalography (EEG) for the measurement process [5]. Among various methods, accurate measurement of driver drowsiness can be achieved with [5]PERCLOS (Percentage of Eye Closure) measurement standing out prominently [6].

The behavioral approach evaluates various actions, including monitoring eye blinks over a specific time interval [7], assessing head pose, steering wheel gripping pressure [8], facial detection, yawning detection [9], and eye gaze. Eye gaze measurement involves the utilization of eye corners and iris centers, employing a linear mapping function to ascertain the direction of the eyes. Pupil center detection relies on a deformable eye based on shape and intensity, utilizing movement decision algorithms [10], with sensitivity to accuracy and diminished performance noted in low-resolution video sequences [11]. The development of the POSIT algorithm (Pose from Orthography and Scaling with Iterations) is facilitated by techniques in head pose estimation and eye gaze [12].

For the driver alerting system of drowsiness, behavioral aspect can be considered most suitable because the evaluation of drowsiness from driver's actions can be achieved with the help of deep learning technology. In this paper restricts our discussion to 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 discuss more about existing proposed driver drowsiness detection system employing deep learning techniques with CNNs. In Section III we present brief overview of CNNs. Our proposed solution is discussed in Section IV. Section V explains how the implementations are performed, to what extent this implementation is conducted, and what the results are. Section VI concludes our paper based on what we discussed.

II. Literature Review/Related Work

III. Deep Learning Background

A Convolutional Neural Network (CNN) is a type of deep learning architecture designed for processing and analyzing visual data, particularly images. CNNs have demonstrated remarkable success in computer vision tasks, including image classification, object detection, and image segmentation. The fundamental building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers. Convolutional Layers have the following characteristics.

First, for convolution layers, convolution operation is the core operation in a convolutional layer. This involves sliding a small window called a filter or kernel across the input image, computing the dot product at each step. The result

is a feature map that captures patterns and features present in different parts of the input. Filters act as feature detectors, recognizing specific patterns like edges, textures, or more complex structures. Multiple filters are applied in parallel, generating multiple feature maps. Each feature map represents the activation of a particular filter across the input. Also stride determines the step size of the filter as it slides across the input. A larger stride reduces the spatial dimensions of the output feature map. Padding is often applied to the input to preserve spatial dimensions and prevent information loss at the borders. There is a concept called receptive field and it refers to the area of the input image that contributes to a particular activation in the feature map. As the convolution operation is applied layer by layer, the receptive field grows, allowing the network to capture hierarchical features.

Secondly, for pooling layers, they downsample the spatial dimensions of the feature maps, reducing computational complexity and controlling overfitting. Max pooling and average pooling are common operations. Max pooling retains the maximum value within a defined window, while average pooling computes the average. Pooling creates spatial hierarchies, where higher-level feature maps represent more abstract and complex features. By progressively reducing spatial dimensions, the network becomes more invariant to translation and distortion, enhancing its ability to recognize patterns in various positions.

Fully Connected Layers are another concept or fundamental building block of CNNs. Fully connected layers with neurons inside follow convolutional and pooling layers. Neurons in these layers are connected to all activations in the previous layer, forming a dense matrix. These layers capture global patterns and relationships in the extracted features. Before entering the fully connected layers, the feature maps are often flattened into a one-dimensional vector. This vector serves as the input to the dense layers, enabling the network to make predictions based on the high-level features learned by the convolutional layers.

Activation Functions such as Rectified Linear Unit (ReLU) introduces non-linearity by setting all negative values to zero. This allows the network to learn complex, non-linear relationships within the data.

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

Since deep learning is a clever techniques it can have a concept called transfer learning. Transfer learning involves using a pre-trained CNN on a large dataset (e.g., ImageNet) as a starting point. The learned features from the pre-trained model can be transferred to a new task, saving training time and enhancing performance, especially when the new task has a limited amount of data.

Data augmentation involves applying random transformations (rotation, scaling, flipping) to the training data. This diversifies the training set, improving the model's ability to generalize to unseen variations. Regularization is a techniques like dropout which are employed to prevent overfitting. Dropout randomly deactivates a fraction of neurons during training, forcing the network to rely on a more robust set of features.

In summary, Convolutional Neural Networks are designed to automatically and adaptively learn hierarchical representations of visual data. The convolutional and pooling layers extract local patterns, while fully connected layers capture global relationships. Activation functions introduce non-linearity, and backpropagation enables the network to learn optimal parameters through training. With advancements like transfer learning and regularization techniques, CNNs have become powerful tools in image analysis, driving breakthroughs in computer vision applications. Understanding the basics of CNNs is essential for practitioners and researchers working in fields where visual data analysis plays a crucial role.

IV. Proposed Work

A.

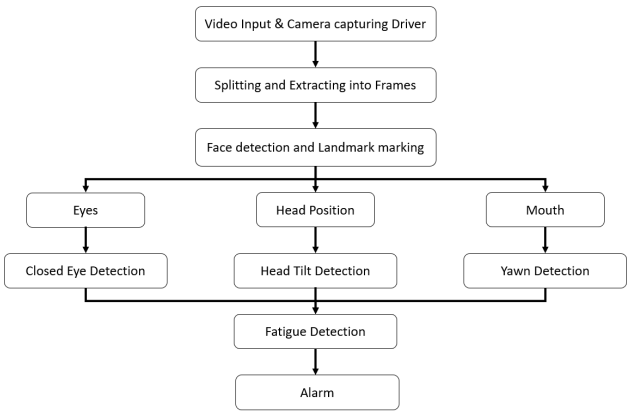

Fig. 1. Example of a figure caption.

V. Implementation and Results

A. Dataset Information

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

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 primary operations within a CNN include convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform the key convolution operation, where small filters slide across the input image, capturing patterns and features. Filters act as feature detectors, identifying specific patterns like edges or textures. The stride and padding parameters control the movement of filters and maintain spatial dimensions. This process generates multiple feature maps, each representing the activation of a distinct filter.

Pooling layers follow convolutional layers, downsampling feature maps to reduce computational complexity and prevent overfitting. Max pooling and average pooling are common operations in this context. Fully connected layers succeed convolutional and pooling layers, incorporating neurons connected to all activations from the previous layer. Flattening the feature maps into one-dimensional vectors precedes these fully connected layers, facilitating global pattern recognition.

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.

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

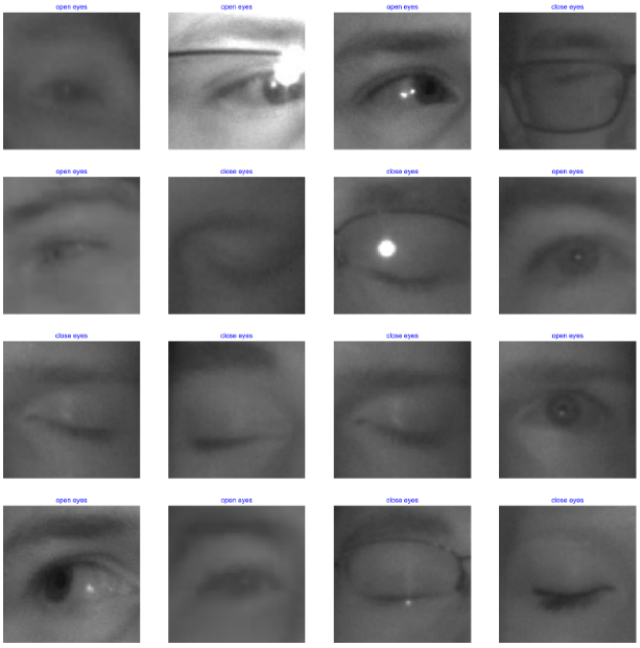

Fig. 2. Example of a figure caption.

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. 3. Example of a figure caption.

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.

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

```
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. 4. Your figure caption goes here.

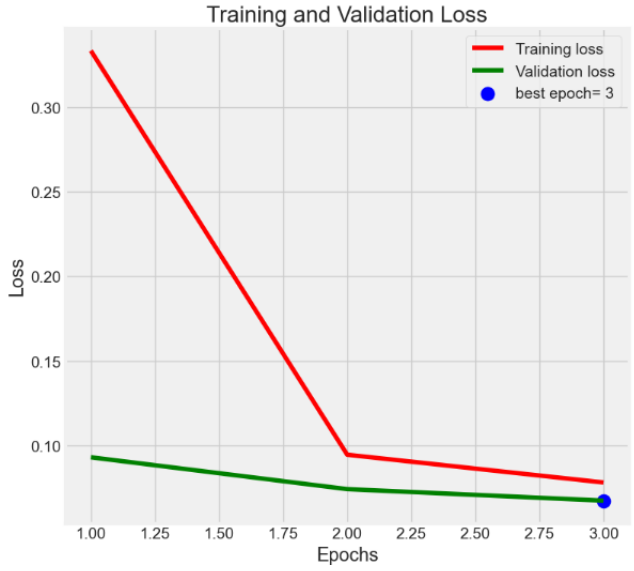

Fig. 5. Example of a figure caption.

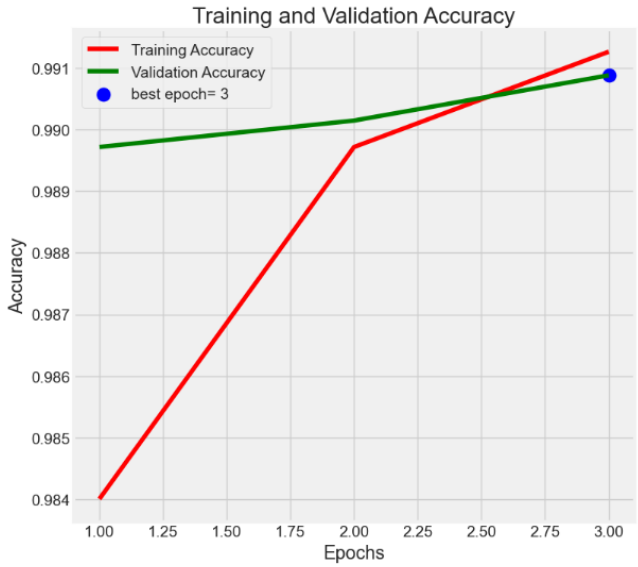

Fig. 6. Example of a figure caption.

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.

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.

VI. Conclusion

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

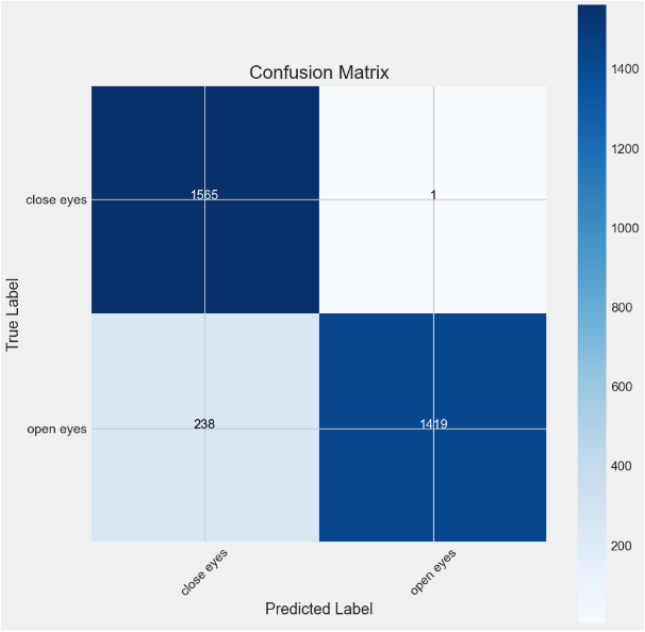

Fig. 7. Example of a figure caption.

	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. 8. Example of a figure caption.

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7].

VII. 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.