# EdgePass: Lightweight Distracted and Drowsiness Driver Detection System

**Wonyoung Jang**
Department of Data Science
Seoul National University
jwy4888@snu.ac.kr

**Sangho Lee**
Department of Data Science
Seoul National University
sangho.lee@snu.ac.kr

**Geo Moon**
Department of English Language and Literature
Seoul National University
moongeo@snu.ac.kr

**Sungwook Son**
Department of Data Science
Seoul National University
sungwookson@snu.ac.kr

## Abstract

Safe driving is an important concern for all drivers. Worldwide, there are a lot of accidents caused by distraction and drowsy driving. This paper presents the system called EdgePass, which detects driver's manual distraction and drowsy driving with convolutional neural network (CNN) based computer vision technology. The system captures the distracted or drowsy driver with an infrared camera regardless of lighting conditions. Using lightweight deep learning architecture with pruning and quantization techniques, this system can perform real-time inference on edge devices. The system is less than 2MB in size and demonstrates 95.3% classification accuracy and 120ms of latency testing on public and private datasets.

## 1 Introduction

This paper proposes Edgepass, a lightweight distracted and drowsy driver detection system. EdgePass comes from the two words Edge and Passenger. EdgePass is a driver alert system that acts as a passenger-user of the edge devices. The purpose of this paper is to contribute to reducing accidents caused by distracted driving by becoming virtual safety personnel.

Distracted driving has become a leading factor of traffic accidents and has been emphasized as a cause of fatal crashes within the decade, along with the marked increase in the use of mobile phones while driving. The AAA Traffic Safety Culture Index found that most drivers (87.5%) believe that distracted driving has become a considerable safety threat, surpassing other traffic-related issues [1]. According to Insurance Information Institute, there were about 3,010 fatal distracted driving induced crashes a year, accounting for about 9% of all fatal crashes [2]. Many recent accident reports indicate an increase in distracted driving, which has caused numerous fatal accidents. Unlike distraction-affected accidents, drowsiness-affected accidents are more difficult to identify; however, while determining the number of drowsy-driving crashes and injuries has been unfeasible, NHTSA estimates that 91,000 police-reported crashes involved drowsy drivers in 2017, and 697 deaths were reportedly caused by drowsiness-affected crashes in 2019 [3].

The National Highway Traffic Safety Administration (NHTSA) defines distracted driving as "any activity that diverts attention from driving, including talking or texting on your phone, eating and drinking, talking to people in your vehicle, fiddling with the stereo, entertainment or navigation system." [4]. KTSA (Korea Transportation Safety Authority), and many other Korean organizations have become more aware of this issue as well. To reduce accident rates, detection using computer vision has been attempted. However, many research and papers suggest either only distracted driver detection or using still images to accurately classify detection type.

## 2 Related Works

Most of the relevant studies have aimed to devise an accurate and robust detection system to improve the performance of identifying unsafe driving and thus to prevent the probable dangerous circumstances. While a few studies have attempted to use CNN on driver's drowsiness detection [5, 6, 7], fatigue detection [8], or distraction detection [9], there is little to no experiments on detecting both distracted and drowsy driving based on CNN. By NHTSA's definition of distracted driving as "anything that takes your attention away from the task of safe driving" [4], this paper attempts to include drowsiness in the broader condition of distraction, and by doing so, aims to define and categorize drowsiness as one of the distracted driver postures. Still, in order to identify drowsiness as a posture with lateral view, temporal information needs to be incorporated.

This paper leans on the works of Baheti et al. [10]. "Detection of Distracted Driver using Convolutional Neural Network" suggests the modified VGG-16 architecture implied with regularization techniques to improve the performance [11]. While major drawback of VGG-16 is the total number of parameters that is nearly 140M, replacing a fully connected layer with $1 \times 1$ convolutional layers saves the number of parameters significantly and thus can be applied to a broad input size (Modified VGG). The result does not only sense the distracted driving but also identify the type of distraction. The performance claims to achieve an accuracy of 96.3% (Table 1).

However, the study is unsuccessful in terms of reaching the lightweight model to make the inference in real-time on the edge device. Considering their unpublished works, it is likely that they struggled to develop a more lightweight model. The performance resulted by implementing these unpublished works was not satisfying as shown in Table 1 (Depthwise VGG). The experiment also points out that it needs to incorporate temporal information in the model to improve the performance.

Table 1: Proposed models by Baheti et al.

| Model | Original VGG | Regularized VGG | Modified VGG | Depthwise VGG |
|---|---|---|---|---|
| Test Accuracy (%) | 94.4 | 96.3 | 95.5 | 45.0 |
| Parameters | 134M | 134M | 15M | 2.3M |

Baheti et al. uses a public dataset for their experiments which is also used in this study [12]. There are ten classes of driver postures: Drive safe, Text left, Talk left, Text right, Talk right, Adjust radio, Drink, Hair and Makeup, Reaching behind, and Talk to passenger; and the respective classes are simulated by various people and cars.

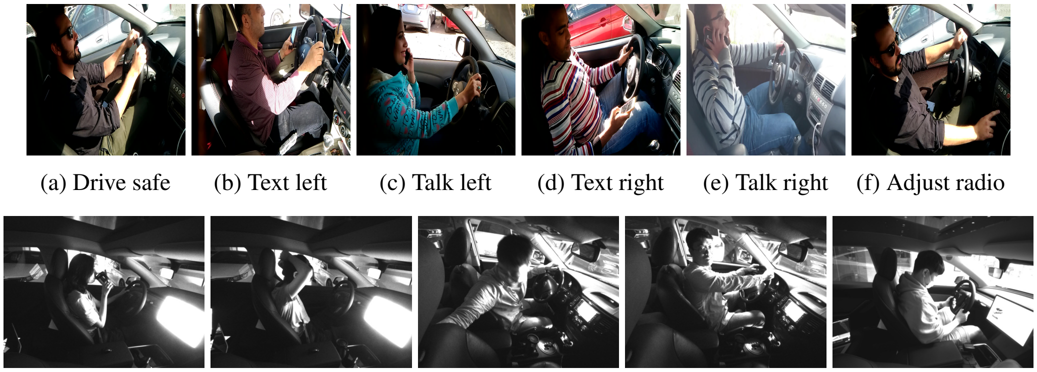
## 3 Materials and Methodologies

### 3.1 Data

One of the biggest technical challenges is to build a high-performing model in low-light conditions. To deal with this problem, this study decides to train the model converting RGB images to gray-scale. It enables the following procedures possible; 1) Shooting of the scenes with an infrared camera, 2) Converting infrared images into visible images. Infrared images become visible as gray-scale to the naked eye, and 3) Putting the gray-scale converted images into the model and run the inference. This was done from the intuition that the RGB image converted to gray-scale and the infrared image converted to gray-scale would be similar. One of the rationales is that the euclidean (L2) distance of pixel values between a gray-scale RGB image and a gray-scale infrared image was similar.

Additionally, the private dataset is created from images taken by an infrared camera. The Dataset is collected from 4 different individuals (3 males and 1 female) and 2 different cars. To increase the diversity of experimental environments, videos are taken from both brightness and darkness conditions. Each frame from the infrared camera was converted into a grayscale image. Then, an additional class, drowsiness, is added. This addition is for preventing the drowsiness class from classified as safe driving. Whether this system could detect drowsiness from side-view or not is another challenge.

As a result, a total of 22,406 images with eleven different classes are used to train and test the model. 12,977 images are from the public dataset presented by Abouelnaga et al. and 9,429 images are from the private dataset [12]. Figure 1 shows the class and example images. First row examples (Figure 1a, 1b, 1c, 1d, 1e, 1f) are from the public dataset, and second row examples (Figure 1g, 1h, 1i, 1j, 1k) are from private dataset.



(a) Drive safe (b) Text left (c) Talk left (d) Text right (e) Talk right (f) Adjust radio

(g) Drink (h) Hair and Makeup (i) Reaching behind (j) Talk to passenger (k) Drowsiness

Figure 1: Dataset examples.

### 3.2 Model

The reference study chooses VGG network as the backbone network [11]. Although the VGG network performs well in various image classification tasks, it is too heavy to be implemented for real-time applications. Therefore, MobileNet version 1 framework is selected to build the model [13]. Pruning is applied in the training phase to make the model more lightweight [14]. MobileNet version 2, 3, and EfficientNet perform unsuccessfully for this task with the pruning-included training scheme [15, 16, 17]. Then, post-quantization is conducted which makes all operations in 8-bit integer. This work makes it possible for the model to operate in the edge device.

### 3.3 Edge Device

In this study, Jetson Nano is used for system implementation. Google's Coral board could be another good option, but depth camera implementation is much easier on Jetson Nano. The system includes the Jetson Nano Development kit and Intel Realsense D435i. Realsense camera includes an RGB camera module and a depth camera. Infrared images are captured using a depth camera. Jetson Nano includes an ARM base CPU and Maxwell GPU capable of running deep learning models. For jetson to operate in a car, Jetson is powered using DC to AC inverter. For both data collection and inference, Realsense camera is connected via USB to retrieve frame images.

To adopt the Realsense camera, Intel Realsense Library (librealsense) is installed. As librealsense is written in C++, integration into python is necessary for model inference. Pyrealsense2, provided by Intel, is a python wrapper for librealsense. The Realsense camera is carefully fixed on the top right of the passenger window. Pyrealsense2 is configured to capture the infrared image frames size of $1920 \times 1080$, 30 frame per second. For the real-time inference, Flask server is used to display our camera image frames and inference result for visualization. Camera installation and Flask server demo are shown in Figure 2.

Then, the alarm system is set up. The frame inference results are accumulated over 2 seconds and if more than 50% of the result is consistently unsafe driving, an alarm triggers to notify the driver to become more attentive. Figure 3 displays the system overview. Jetson Nano captures the video frame input from the driver and processes the frames with the pruned and quantized model. If the alarming threshold is reached, EdgePass alarms the driver through a speaker.

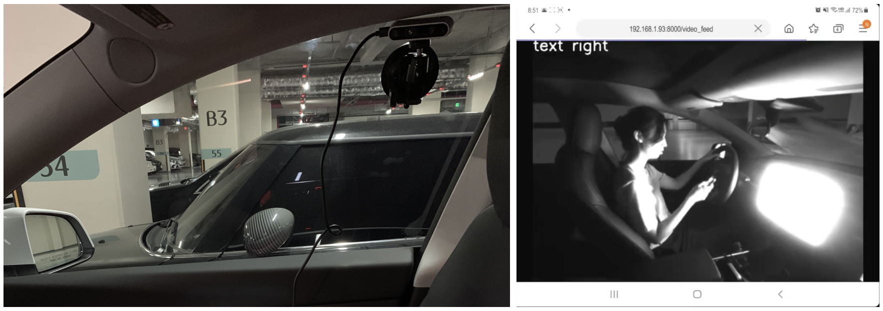


Figure 2: Camera installation and Flask server demo.

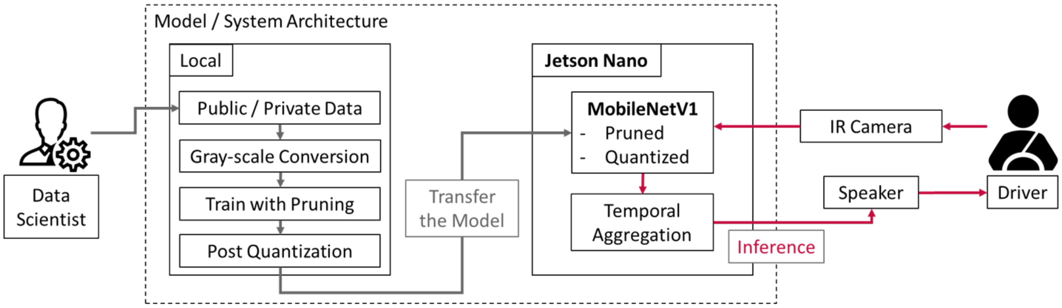


Figure 3: EdgePass system overview.

## 4 Results

The model is trained with 22,406 gray-scale converted images; 12,977 images from public dataset and 9,429 images from private dataset. The following hyperparameters; epochs = 20, optimizer = Adam, loss function = sparse categorical crossentropy, batch size = 128, learning rate (lr) = 1e-3 (lr time based decay after epoch = 10), initial sparsity = 0.5, final sparsity = 0.8; are used for training. Tensorflow and TensorflowLite are selected as the deep learning framework.

Then, the model is tested with 7,481 gray-scale converted images; 4,331 images from public dataset and 3,150 images from private dataset. The results are shown in Table 2. EdgePass shows decent performance for the task and maintains its performance even after the quantization. The model size is decreased by about 70% after quantization.

Table 2: EdgePass test accuracy.

| Model | EdgePass (pruned) | EdgePass (pruned & quantized) |
|---|---|---|
| Test Accuracy (%) | 95.7 | 95.3 |
| Model Size (MB) | 5.31 | 1.66 |

To evaluate the performance, comparison has been made between EdgePass and the model from the reference study [10]. In order to compare these two models in the same environment, the VGG model is quantized and transferred to Jetson Nano. The results are shown in Table 3. Compared to the previous work (95.4%), EdgePass shows almost the same performance (95.3%) while it outperforms in model size (1.66MB) and latency (120ms). It verifies that EdgePass is adequate for the real-time application. The proposed temporal aggregating alarm system also works well (Figure 4). A demo video is available at this link (http://bit.ly/EdgePass).

Table 3: EdgePass and VGG comparison on Jetson Nano.

| Model | Modified VGG (quantized) | EdgePass (pruned & quantized) |
|---|---|---|
| Test Accuracy (%) | 95.4 | 95.3 |
| Model Size (MB) | 26.68 | **1.66** |
| Latency (ms) | 2,200 | **120** |

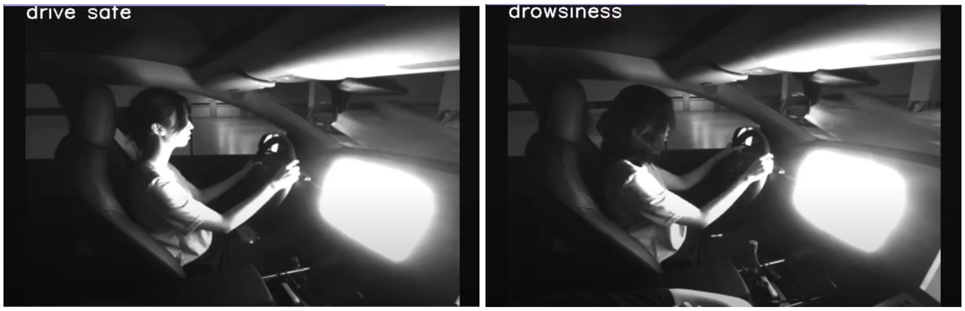


Figure 4: The real-time inference examples from EdgePass.

## 5 Conclusion

This paper presents EdgePass, the real-time side-view driver distraction and drowsiness detection model with an alarm system. Edgepass is lightweight, performs well, is robust in any light condition, and considers temporal information. This system also shows that side-view images are capable of detecting drowsiness when considering temporal information. However, there is still plenty of room to improve this system. First, the current system tends to be sensitive to the camera angle. Another limitation is that the system is hard to deal with corner cases. For instance, if the driver looks back while driving backward, the system identifies it as ’reaching behind’ and alarms. To prevent such confusion, Inertial measurement unit (IMU) sensor can be used. Another suggestion is aggregating frontal-view images and depth information to improve the system performance. Also reducing depth information into 2D images due to data shortage leads to the loss of important information; a more robust model could be implemented using depth information. As Shuai et al. suggests, incorporating RGB stream and depth stream together at the final stage [18] could be an alternative to improve model. Lastly, a more practical duration or threshold of the alarm could be found through future experiments. If the above constraints could be solved, a more practical and effective distracted and drowsiness driver detection system could be developed.

# References

[1] AAA Foundation for Traffic Safety. 2017 traffic safety culture index (technical report), 2018.

[2] Insurance Information Institute. 2017 traffic safety culture index (technical report), 2018.

[3] National Highway Traffic Safety Administration. Drowsy driving, 2019.

[4] National Highway Traffic Safety Administration. Distracted driving, 2019.

[5] Maryam Hashemi, Alireza Mirrashid, and Aliasghar Beheshti Shirazi. Driver safety development real time driver drowsiness detection system based on convolutional neural network, May 2021.

[6] Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, Moez Krichen, and Kamel Barkaoui. Driver drowsiness detection model using convolutional neural networks techniques for android application. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 237–242, 2020.

[7] Muhammad Shakeel, Nabit Bajwa, Ahmad Anwaar, Anabia Sohail, Asifullah Khan, and Haroon Ur Rashid. *Detecting Driver Drowsiness in Real Time Through Deep Learning Based Object Detection*, pages 283–296. 05 2019.

[8] Zuopeng Zhao, Nana Zhou, Lan Zhang, Hualin Yan, Yi Xu, and Zhongxin Zhang. Driver fatigue detection based on convolutional neural networks using em-cnn, Nov 2020.

[9] Munif Alotaibi and Bandar Alotaibi. Distracted driver classification using deep learning. *Signal, Image and Video Processing*, 14(3):617–624, November 2019.

[10] Bhakti Baheti, Suhas Gajre, and Sanjay Talbar. Detection of distracted driver using convolutional neural network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1145–11456, 2018.

[11] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[12] Yehya Abouelnaga, Hesham M Eraqi, and Mohamed N Moustafa. Real-time distracted driver posture classification. *arXiv preprint arXiv:1706.09498*, 2017.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[16] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[17] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[18] Xian Shuai, Yulin Shen, Yi Tang, Shuyao Shi, Luping Ji, and Guoliang Xing. Millieye: A lightweight mmwave radar and camera fusion system for robust object detection. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, IoTDI '21, page 145–157, New York, NY, USA, 2021. Association for Computing Machinery.