Distracted Driver Classification for Low-computing Devices using CNN

1st Long Nguyen
Computer Science
Penn State Harrisburg
Middletown, USA
lhn5032@psu.edu

2nd Sukmoon Chang
Computer Science
Penn State Harrisburg
Middletown, USA
sukmoon@psu.edu

***Abstract*—Distracted driving is one of the major causes of traffic accidents. To help more drivers become aware of their driving behavior, this paper proposes a distracted driver classifier for low-computing devices. The classifier is based on a simple Convolutional Neural Network (CNN) with Depthwise Separable convolution. The classifier was evaluated with the State Farm Distracted Driver Dataset and achieved a 99.51% accuracy while maintaining a speed of 23.72 frames per second (FPS) on the CPU. Our proposed classifier will allow more cars to implement distracted driver detection systems, and thus, will help reduce the number of traffic accidents overall.**

***Index Terms*—Distracted driver classifier, Low-computing devices, Convolutional Neural Network, Depthwise Separable Convolution,**

I. Introduction

According to the National Highway Traffic Safety Administration (NHTSA), 38,824 people died and 2.28 million people injured due to motor vehicle traffic crashes in 2020 [1]. A more elaborated report by the NHTSA states that distracted driving is responsible for 8.1% (3,142) of all traffic fatalities and about 324,652 injured people [2].

The NHTSA defined 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 — anything that takes your attention away from the task of safe driving” [3]. There are three main types of distraction in driving according to the Center for Disease Control and Prevention (CDC): *Visual distraction, Manual distraction,* and *Cognitive distraction*. Visual distraction refers to the driver taking off his/her eyes from the road. Manual distraction refers to the driver taking his/her hands off the steering wheel. Cognitive distraction refers to the driver taking his/her mind off the task of driving [4].

Many distraction detection systems have been implemented in cars to help reduce these types of distractions. For example, in visual distraction, cameras were installed in the vehicle to keep track of the driver’s eyes state and the position of his/her head [5]. In manual distraction, sensor mats were installed in steering wheel to determine whether the driver hands are on the wheel or not [6]. In cognitive distraction, methods used in visual distraction and manual distractions can be combined with other types of sensors to help detect cognitive distraction [7, 8].

In our paper, we focus on improving an existing distracted driver classifier from Nguyen *et al.*’s work [9]. The main contributions of our paper include (1) showing multiple modifications of an existing classifier and (2) improving the best modification with depthwise separable convolution. Our contributions allow distracted driver detection systems to be implemented efficiently on low-computing devices.

II. Literature Review

In this section, the paper will introduce CNN-based methods that were implemented to make distracted driver classification possible in real-time on the CPU. Three main topics for those methods will be discussed in details: *A. Simple CNN and Global Average Pooling, B. Simple CNN with preprocessing image techniques*, and *C. MobileVGG.*

A. Simple CNN and Global Average Pooling

One approach to make classification faster is to make the CNN model as simple as possible. For example, research done by the University of Ulsan, Korea, proposes a simple CNN that can efficiently recognize the distracted behavior of drivers using a camera. The main contribution of this work is to reduce the number of parameters in the neural network, while maintain high accuracy and acceptable speed. This CNN model achieved high accuracy (99.51%), consisted of only 0.6 milions parameters, while maintaining an acceptable inference speed (14.26 FPS on 3.4 GHz CPU) [9]. The paper utilize Global Average Pooling to reduce the number of parameters, and the simpleness of CNN to achieve such result. However, the downside of this method is that Global Average Pooling only work well if the feature map before is relatively small and contains mostly important information. This means the network has to be deeper than necessary, and thus would require more computations to process many layers.

B. Simple CNN and preprocessing image techniques

Simple CNN is good, but if the model is too simple, some information can be lost. For example, Leekha *et al.* suggested a very simple CNN that takes in the image input size of 100x100x3 [10]. This means the input image has to be resized significantly before it can be evaluated by this model, implying information loss. To account for this loss, the authors used an image segmentation algorithm called GrabCut, which removes all unnecessary background noises in an image and retains only important information in the image. After that, the image will be fitted into the CNN for training or evaluation. For another example, Quin *et al.* built a simple CNN that has decreasing filter size [11]. The main purpose of this approach is to reduce the number of parameters in the network. Therefore, to ensure the accuracy does not decrease when the number of parameters in the network decreases, Histogram of Oriented Gradients (HOG) technique was used to preprocess the images. HOG's purpose is similar to Grabcut, which is to remove background noises and keep only important features in the image.

C. MobileVGG

MobileVGG is a CNN model for distracted driver classification that is based on the *Depthwise Separatable Convolution* and *Visual Geometry Group (VGG)* architecture [12]. Depthwise Separable Convolution is an alternative way of calculating convolution in CNN, which significantly reduce the number of computations compared to the standard convolution computation. On the other hand, VGG architecture promotes a simple and deep neural network when designing a model, and is well-known for image classification problems. MobileVGG utilizes the Depthwise Separable Convolution to increase the speed, while using the VGG architecture to achieve high accuracy. MobileVGG was able to achieve 99.75% accuracy on the StateFarm distracted driver dataset [14] while achieved a speed of 35 FPS on CPU (Intel Xeon Processor)

From analyzing the above related works, we proposes a simple CNN model with Depthwise Separable Convolution that is simple, fast, and achieves high accuracy.

III. Methodology

Our methodology is an attempt to improve one of the methods described above. We want to improve the simple neural network that uses the Global Average Pooling by Nguyen *et al.* [9], as described in section II.A. We chose it because the model is simple and easy to understand. The model also does not use any image preprocessing algorithms but is still able to achieve high accuracy with acceptable speed.

There are two stages to our methodology. The first is about finding the best modification of the model through trial and error, based on the Global Average Pooling layer. The second is about improving the best model with Depthwise Separable Convolution.

A. Original Model

The model we picked contains 4 convolutional blocks, each block contains an average pooling layer. After the blocks, there are two more convolutional layers, followed by the Global Average Pooling layer, and then the output layers [9]. Table I shows the details of the original model.

TABLE I
ORIGINAL MODEL ARCHITECTURE DETAILS

Layer Types	**Output Shape**
Block1	
Conv1, $7 \times 7 \times 16$	$224 \times 224 \times 16$
Conv2, $7 \times 7 \times 16$	$224 \times 224 \times 16$
Pool1, $2 \times 2 \times 16$	$112 \times 112 \times 16$
Block2	
Conv3, $5 \times 5 \times 32$	$112 \times 112 \times 32$
Conv4, $5 \times 5 \times 32$	$112 \times 112 \times 32$
Pool2, $2 \times 2 \times 32$	$56 \times 56 \times 32$
Block3	
Conv5, $3 \times 3 \times 64$	$56 \times 56 \times 64$
Conv6, $3 \times 3 \times 64$	$56 \times 56 \times 64$
Pool3, $2 \times 2 \times 64$	$28 \times 28 \times 64$
Block4	
Conv7, $3 \times 3 \times 128$	$28 \times 28 \times 128$
Conv8, $3 \times 3 \times 128$	$28 \times 28 \times 128$
Pool4, $2 \times 2 \times 128$	$14 \times 14 \times 128$
Conv9, $3 \times 3 \times 256$	$14 \times 14 \times 256$
Conv10, $3 \times 3 \times 10$	$14 \times 14 \times 10$
GlobalAveragePooling	$1 \times 1 \times 10$
Softmax	$1 \times 1 \times 10$

B. Modify the Architecture

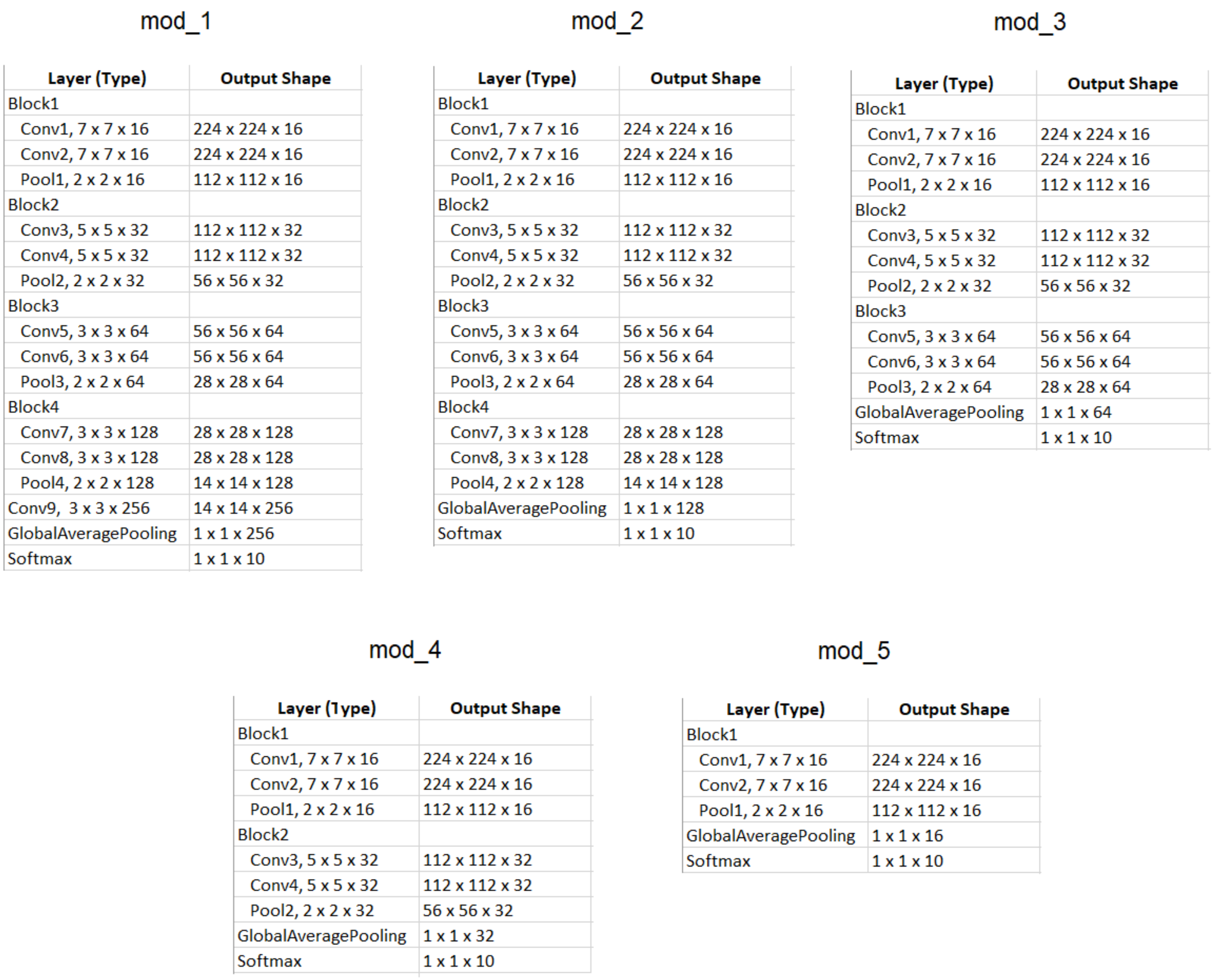
We made 10 modifications to the original model to understand how it works. We performed 2 sets of experiments (5 modifications each), one with the Global Average Pooling layer and one without the Global Average Pooling layer. For each set of experiments, we removed the components in the model iteratively, starting from the two convolutional layers at the end of the network, and working our way backward to the convolutional blocks. We did this to see the impact Global Average Pooling has on the network since the paper stated that Global Average Pooling is crucial for this model because it helps reduce the number of parameters in the network. From there, the best modification is picked to be improved further. Figure 1 shows the details of modifications 1 to 5, with the Global Average Pooling layer. Modification 6 to 10 is the repetition of 1 to 5 but without the Global Average Pooling layer.

C. More Improvement with Depthwise Separable Convolution

After we picked the best modification of the model, we uses *Depthwise Separable Convolution* to further improve it.

1) Standard Convolution: Convolutional Neural Network by default uses standard convolution. To calculate a convolution using standard convolution, the network takes in the input of size $f \times f \times m$, where $f \times f$ is the size of one input channel, and m is the number of input channels. It then applies a filter of size $k \times k \times m$, where $k \times k$ is the kernel size. After slide a filter through the entire input, we got an output of size $g \times g$, where $g \leq f$. If we apply N filters, then we will get an output of size $g \times g \times N$. Figure 2 visualizes this process.

2) Depthwise Separable Convolution: Depthwise Separable Convolution was first mentioned by Sifre *et al.*[13]. It is an alternative way of computing convolution. Unlike standard convolution, it has two stages:



mod_1

Layer (Type)	Output Shape
Block1	
Conv1, 7 x 7 x 16	224 x 224 x 16
Conv2, 7 x 7 x 16	224 x 224 x 16
Pool1, 2 x 2 x 16	112 x 112 x 16
Block2	
Conv3, 5 x 5 x 32	112 x 112 x 32
Conv4, 5 x 5 x 32	112 x 112 x 32
Pool2, 2 x 2 x 32	56 x 56 x 32
Block3	
Conv5, 3 x 3 x 64	56 x 56 x 64
Conv6, 3 x 3 x 64	56 x 56 x 64
Pool3, 2 x 2 x 64	28 x 28 x 64
Block4	
Conv7, 3 x 3 x 128	28 x 28 x 128
Conv8, 3 x 3 x 128	28 x 28 x 128
Pool4, 2 x 2 x 128	14 x 14 x 128
Conv9, 3 x 3 x 256	14 x 14 x 256
GlobalAveragePooling	1 x 1 x 256
Softmax	1 x 1 x 10

mod_2

Layer (Type)	Output Shape
Block1	
Conv1, 7 x 7 x 16	224 x 224 x 16
Conv2, 7 x 7 x 16	224 x 224 x 16
Pool1, 2 x 2 x 16	112 x 112 x 16
Block2	
Conv3, 5 x 5 x 32	112 x 112 x 32
Conv4, 5 x 5 x 32	112 x 112 x 32
Pool2, 2 x 2 x 32	56 x 56 x 32
Block3	
Conv5, 3 x 3 x 64	56 x 56 x 64
Conv6, 3 x 3 x 64	56 x 56 x 64
Pool3, 2 x 2 x 64	28 x 28 x 64
Block4	
Conv7, 3 x 3 x 128	28 x 28 x 128
Conv8, 3 x 3 x 128	28 x 28 x 128
Pool4, 2 x 2 x 128	14 x 14 x 128
GlobalAveragePooling	1 x 1 x 128
Softmax	1 x 1 x 10

mod_3

Layer (Type)	Output Shape
Block1	
Conv1, 7 x 7 x 16	224 x 224 x 16
Conv2, 7 x 7 x 16	224 x 224 x 16
Pool1, 2 x 2 x 16	112 x 112 x 16
Block2	
Conv3, 5 x 5 x 32	112 x 112 x 32
Conv4, 5 x 5 x 32	112 x 112 x 32
Pool2, 2 x 2 x 32	56 x 56 x 32
Block3	
Conv5, 3 x 3 x 64	56 x 56 x 64
Conv6, 3 x 3 x 64	56 x 56 x 64
Pool3, 2 x 2 x 64	28 x 28 x 64
GlobalAveragePooling	1 x 1 x 64
Softmax	1 x 1 x 10

mod_4

Layer (Type)	Output Shape
Block1	
Conv1, 7 x 7 x 16	224 x 224 x 16
Conv2, 7 x 7 x 16	224 x 224 x 16
Pool1, 2 x 2 x 16	112 x 112 x 16
Block2	
Conv3, 5 x 5 x 32	112 x 112 x 32
Conv4, 5 x 5 x 32	112 x 112 x 32
Pool2, 2 x 2 x 32	56 x 56 x 32
GlobalAveragePooling	1 x 1 x 32
Softmax	1 x 1 x 10

mod_5

Layer (Type)	Output Shape
Block1	
Conv1, 7 x 7 x 16	224 x 224 x 16
Conv2, 7 x 7 x 16	224 x 224 x 16
Pool1, 2 x 2 x 16	112 x 112 x 16
GlobalAveragePooling	1 x 1 x 16
Softmax	1 x 1 x 10

Fig. 1. Model modification 1 to 5: With GlobalAveragePooling

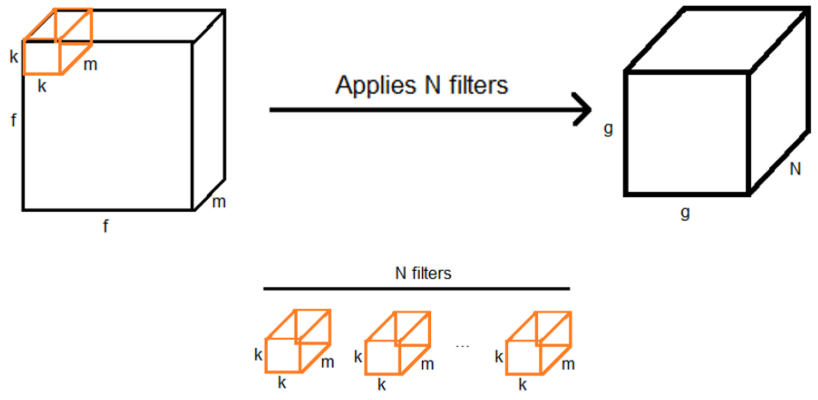


Fig. 2. Visualization of Standard Convolution

a) Depthwise convolution: Depthwise convolution takes in the input of size $f \times f \times m$. But instead of using a filter of size $k \times k \times m$, it uses m filters of size $k \times k$. Each of those m filters will be applied to each input channel. The output will be of size $g \times g \times m$ [12]. This process is visualized in Figure 3.

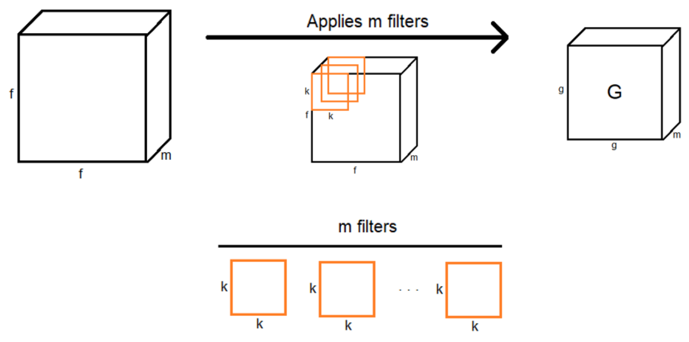


Fig. 3. Visualization of depthwise convolution

b) Pointwise convolution: The output of Depthwise convolution will be used as the input for pointwise convolution, which is $g \times g \times m$. Filter of size $1/times1/timesm$ will be used to combine the input channels. After applying such filter, the output will have the size of $g \times g$. IF we apply N such filters, the output will be $g \times g \times N$ [12]. Figure 4 visualizes this process.

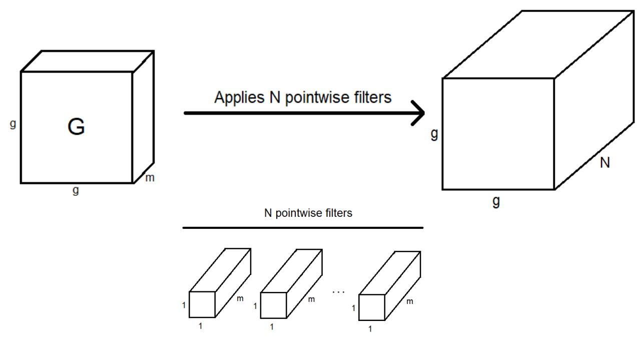


Fig. 4. Visualization of pointwise convolution

3) Standard convolution vs Depthwise Separable Convolution: We will calculate the number of computations each method used and compare them. Since addition is cheap, we will only consider the number of multiplications as the number of computations.

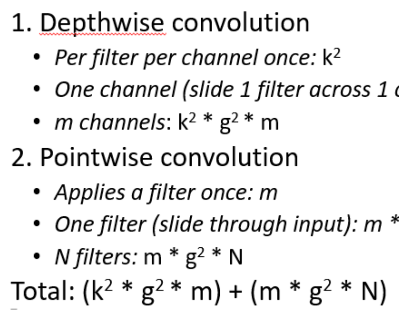
a) Standard Convolution: The number of multiplication in a standard convolution is as follows [12]:

Applied a filter once (without sliding through the input): $k^2 \times m$

Applied a filter (slide through the input): $k^2 \times m \times g^2$

Applied N filters: $k^2 \times m \times g^2 \times N$

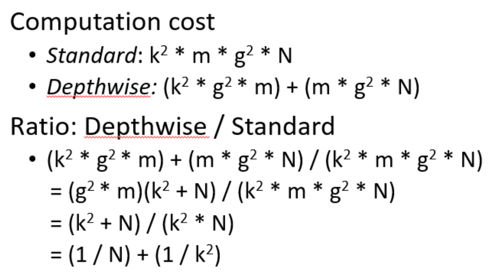
b) Depthwise Separable Convolution: The number of multiplication in a depthwise separable convolution is as follows [12].



1. Depthwise convolution
   - *Per filter per channel once:* k^2
   - *One channel (slide 1 filter across 1 channel):* $k^2 * g^2$
   - *m channels*: $k^2 * g^2 * m$
2. Pointwise convolution
   - *Applies a filter once: m*
   - *One filter (slide through input): m * g^2*
   - *N filters:* m * g^2 * N

Total: ($k^2 * g^2 *$ m) + (m * g^2 * N)

c) Computation Ratio: The computation ratio of depthwise convolution is as follows [12]:



Computation cost
- *Standard*: $k^2 * m * g^2 * N$
- *Depthwise:* $(k^2 * g^2 * m) + (m * g^2 * N)$

Ratio: Depthwise / Standard
- $(k^2 * g^2 * m) + (m * g^2 * N) / (k^2 * m * g^2 * N)$
  $= (g^2 * m)(k^2 + N) / (k^2 * m * g^2 * N)$
  $= (k^2 + N) / (k^2 * N)$
  $= (1 / N) + (1 / k^2)$

For example, if the number of filters (N) is 256, and the filter size (k) is 3, then the ratio would be:

$$\frac{1}{N} + \frac{1}{k^2} = \frac{1}{256} + \frac{1}{9} = 0.444 + 0.111 = 0.115$$

This implies Depthwise Separable Convolution is at least 8.6 times faster than Standard Convolution in this example. Therefore, we will use Depthwise Separable Convolution to improve our best modification of the original model.

IV. Results

A. Dataset

State Farm Distracted Driver Dataset was used to evaluate our results [14]. The dataset consists of 22,424 colored images with a resolution of 640 $\times$ 480. The dataset consists of 10 classes, c0 to c9, where c0 is safe driving, and c1 to c9 is distracted driving behaviors. Figure 5 shows the 10 classes in detail.

Furthermore, the dataset is divided into 3 main parts: Train data (60% - 13458 images), validation data (20% - 4483 images), and test data (20% - 4483 images). We use train data to train our models, and validation data to validate our models. The test data is only used after we came up with our proposed model to avoid bias.

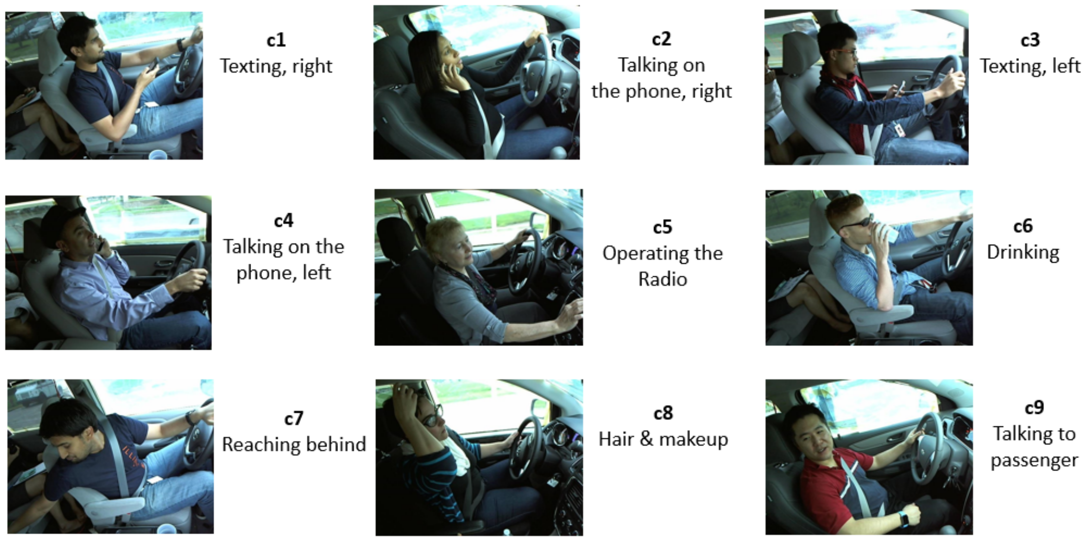


Fig. 5. Illustration of 10 classes in State Farm Distracted Driver Dataset

B. Environmental Setup

We evaluated our models on Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (upto 3.4 GHz). Additionally, we follow the same basic configurations from the work of Nguyen *et al.* [9] when training our models. Those configurations are as follows: the number of epochs is 200, the learning rate is 10^{-4}, the batch size is 16, and the optimization method is Adam optimization method. However, unlike the work of Nguyen *et al.*, we did not use any image augmentation techniques or batch normalization, since their work left out those details.

C. Original Model results

Table II shows the results of the original model with and without the Global Average Pooling layer. Notice that there is not a significant difference when removing the Global Average Pooling layer, except that the number of parameters has increased. This is because the purpose of the Global Average Pooling layer is to reduce the number of parameters, and thus when it is removed, the number of parameters increases.

D. Modifications results

Table III shows the results of our modifications with the Global Average Pooling layer. Notice that the inference time decreases as we remove each component of the model. This is because fewer components imply fewer computations. Furthermore, the accuracy stays consistent at the beginning of the modification process but dramatically decreased at the end. This is because the Global Average Pooling layer replaces the fully connected layer with the average of each channel of the layer before it. The purpose of the Global Average Pooling layer is to reduce the number of parameters in a network. However, if there are a lot of values in a channel, then the average value will fail to accurately represent that channel, and thus, the accuracy of the model will be reduced. This fact is clearly reflected in mod_5, where the layer before the Global Average Pooling layer consisted of 112×112 (12544) values in each channel.

Table IV shows the results of our modifications without the Global Average Pooling layer. Notice that the inference time decreases as we remove each component of the model. Again, this is because fewer components imply fewer computations. However, unlike our modifications with the Global Average Pooling layer, the accuracy stays consistent for these modifications. This is because each channel from the layer before the fully connected layer does not get simplified with an average value. This implies without Global Average Pooling Layer, a simple and shallow CNN is enough to classify distracted drivers with high accuracy and high speed.

E. Depthwise Separable Convolution results

From Table III and Table IV, we conclude that mod_10 is the best modification. Table V shows the result of mod_10 but with depthwise separable convolution on validation and test data. Our new model was able to achieve 99.51% and a speed of 23.72 FPS on the test data while consisting of about 2 million parameters. We also evaluated our original model on test data, and the original model achieved an accuracy of 99.44 % with a speed of 12.52 FPS. This implies our model is almost as twice as fast as the original model while maintaining high accuracy. Table VI shows the details of our new model.

V. Conclusion

This paper proposed an improvement from the work of Nguyen *et al.* [9]. Our proposed model is a very simple Convolutional Neural Network that uses Depthwise Separable Convolution. We made 10 modifications to the original models using trial and error, based on the Global Average Pooling layer. After that, we picked out the best modification and further improve it with Depthwise Separable Convolution. Our

TABLE II
ORIGINAL MODEL RESULTS WITH AND WITHOUT GLOBAL AVERAGE POOLING

Original Model	Accuracy (%)	Inference time (ms / image)	Number of parameters
With Global Average Pooling layer	99.31	78.96	648,584
Without Global Average Pooling layer	99.55	79,19	668,084

TABLE III
MODIFICATION RESULTS WITH GLOBAL AVERAGE POOLING LAYER ON VALIDATION DATASET

Modifications	Accuracy (%)	Inference time (ms / image)
mod_1	99.33	77.84
mod_2	99.31	76.06
mod_3	99.53	70.27
mod_4	94.89	63.57
mod_5	72.65	43.72

TABLE IV
MODIFICATION RESULTS WITHOUT GLOBAL AVERAGE POOLING LAYER ON VALIDATION DATASET

Modifications	Accuracy (%)	Inference time (ms / image)
mod_6	99.60	78.51
mod_7	99.40	76.29
mod_8	99.62	69.37
mod_9	99.31	63.80
mod_10	**99.26**	**44.39**

model is about two times faster than the original model while maintaining high accuracy. The number of parameters in our model is about three times more than the number of parameters in the original model. However, this is an acceptable result since 2 million parameters is still very small compares to other popular models [12]. With the accuracy and speed of our model, it can be deployed to low-computing devices for real-time applications such as alerting drivers when they are distracted. It can also be used to create reports of the drivers' driving behaviors to help the driver be more informed of their driving behaviors.

TABLE V
RESULT OF MOD_10 WITH DEPTHWISE SEPARABLE CONVOLUTION

Data	Accuracy (%)	Inference time (ms / image)
Validation Data	99.49	41.71
Test Data	99.51	42.16

TABLE VI
PROPOSED ARCHITECTURE DETAILS

Layer Types	Output Shape
Block1	
SeparableConv1, $7 \times 7 \times 16$	$224 \times 224 \times 16$
SeparableConv2, $7 \times 7 \times 16$	$224 \times 224 \times 16$
Pool1, $2 \times 2 \times 16$	$112 \times 112 \times 16$
Dense	$1 \times 1 \times 200704$
Softmax	$1 \times 1 \times 10$

REFERENCES

[1] Stewart, T. (2022, March). Overview of motor vehicle crashes in 2020 (Report No. DOT HS 813266). National Highway Traffic Safety Administration.

[2] National Center for Statistics and Analysis. (2022, May). Distracted driving 2020 (Research Note. Report No. DOT HS 813 309). National Highway Traffic Safety Administration.

[3] "Distracted driving," NHTSA. [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving. [Accessed: 10-Oct-2022].

[4] "Distracted driving," Centers for Disease Control and Prevention, 26-Apr-2022. [Online]. Available: https://www.cdc.gov/transportationsafety/Distracted_Driving/index.html. [Accessed: 10-Oct-2022].

[5] A. S. Le, T. Suzuki, and H. Aoki, "Evaluating driver cognitive distraction by eye tracking: From Simulator to driving," Transportation Research Interdisciplinary Perspectives, vol. 4, p. 100087, 2020.

[6] "Hands off detection (HOD) sensing system - IEE smart sensing solutions." [Online]. Available: https://iee-sensing.com/app/uploads/2021/08/iee-hod-hands-off-detection-sensing-system-factsheet.pdf. [Accessed: 23-Oct-2022].

[7] A. Kashevnik, R. Shchedrin, C. Kaiser and A. Stocker, "Driver Distraction Detection Methods: A Literature Review and Framework," in IEEE Access, vol. 9, pp. 60063-60076, 2021, doi: 10.1109/ACCESS.2021.3073599.

[8] Q. Sun, C. Wang, Y. Guo, W. Yuan, and R. Fu, "Research on a Cognitive Distraction Recognition Model for Intelligent Driving Systems Based on Real Vehicle Experiments," Sensors, vol. 20, no. 16, p. 4426, Aug. 2020, doi: 10.3390/s20164426.

[9] D.-L. Nguyen, M. D. Putro, and K.-H. Jo, "Distracted driver recognizer with simple and efficient convolutional neural network for real-time system," 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021.

[10] M. Leekha, M. Goswami, R. R. Shah, Y. Yin and R. Zimmermann, "Are You Paying Attention? Detecting Distracted Driving in Real-Time," 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), 2019, pp. 171-180, doi:10.1109/BigMM.2019.00-28.

[11] B. Qin, J. Qian, Y. Xin, B. Liu, and Y. Dong, "Distracted driver detection based on a CNN with decreasing filter size," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6922–6933, 2022.

[12] B. Baheti, S. Talbar, and S. Gajre, "Towards computationally efficient and realtime distracted driver detection with mobilevgg network," IEEE Transactions on Intelligent Vehicles, vol. 5, no. 4, pp. 565–574, 2020.

[13] L. Sifre Supervisor and S. Mallat, "Ecole polytechnique, CMAP rigid-motion scattering for image classification," 2014. [Online].Available: http:www.di.ens.fr/data/publications/papers/phd_sifre.pdf

[14] "State Farm distracted driver detection," Kaggle. [Online]. Available: https://www.kaggle.com/competitions/state-farm-distracted-driver-detection/data. [Accessed: 10-Oct-2022].