IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 26, 2020, accepted October 28, 2020, date of publication November 3, 2020, date of current version November 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035393

An Adaptive Batch-Image Based Driver Status Monitoring System on a Lightweight GPU-Equipped SBC

WHUI KIM[1], YOUL-KYEONG LEE[2], (Member, IEEE), WOO-SUNG JUNG[1], (Member, IEEE), DAESEUNG YOO[1], DO-HYUN KIM[1], AND KANG-HYUN JO[2], (Senior Member, IEEE)

[1]Intelligent Robotics Research Division, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea
[2]Graduate School of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, South Korea

Corresponding author: Kang-Hyun Jo (acejo@ulsan.ac.kr)

This work was supported in part by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean Government (The Development of Smart Context-Awareness Foundation Technique for Major Industry Acceleration) under Grant 20ZS1200, and in part by the Uvalsan Metropolitan City (Development of Smart HSE System and Digital Cockpit System based on ICT Convergence for Enhanced Major Industry) under Grant 20AS1100.

ABSTRACT Most Driver Status Monitoring (DSM) systems consist of spatial features extractions and temporal status recognition in sequence. Extracting the spatial features, which include driver facial behavior such as eye closing and mouth opening, generally requires considerable computation. It causes DSM systems to lose the valuable instantly-occurring driver facial information during real-time processing. Loss of facial information affects the accuracy of the system, and its impact is more severe on restricted computing resources. To solve this problem, this paper proposes an Adaptive Batch-Image (ABI) based DSM (ABI-DSM) system. The ABI enables the DSM system to use images captured in real-time while the DSM process previous input images. For real-time operation on a lightweight GPU-equipped Single-Board Computer (SBC), the ABI-DSM system is designed as follows. First, the system uses the driver's facial behavior to reduce the dimension of the time-series data for recognizing the status of the driver. The second, detection and tracking of driver's faces are not used for facial behavior recognition. Also, the system works with PydMobileNet, which has lower parameters and FLOPs than MobileNetV2, for facial behavior recognition. Experiments show that the ABI-DSM systems based on MobileNetV2 and PydMobileNet perform better than others in terms of both FPS and Precision. In particular, the PydMobileNet-based ABI-DSM system outperforms competitors when the size of Batch-Image is over six images.

INDEX TERMS MobileNet, PydMobileNet, long short-term memory, driver status monitoring, drowsy, fatigue, distraction, single board computer.

I. INTRODUCTION

To reduce traffic accident casualties caused by distracted and drowsy driving, many researchers have published works investigating vision-based driver assistance systems. Driver status recognition, along with vehicle, pedestrian, sign, and lain line detection, is one of the actively studied areas in the automotive industry. Vision-based driver monitoring systems are not intrusive because they use a mounted camera, unlike systems based on physiological measures that require the attachment of Electroencephalography (EEG) and Electrocardiography (ECG) sensors to the driver's body [1]. Although wearable devices with EEGs or ECGs recently have been released, they are generally not capable of sufficiently reducing vehicle-induced interference that influences data quality [2]. Vision-based systems also have the advantage of using driver facial behaviors directly related to driver's status in contrast to in-vehicle sensor-based systems using indirect information, such as steering wheel movement or the standard deviation of the lane position. Thus, in-vehicle sensor-based and physiological signal-based systems can be influenced by external factors, while vision-based systems are rarely affected by such factors [3]–[5]. The research and development of vision-based driver monitoring systems

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

have become more active owing to the recent growth of the compact camera, vehicular ISP (Image Signal Processor), and mobile GPU (Graphics Processing Unit) markets [6].

Driver monitoring systems had traditionally used hand-crafted descriptors and Machine Learning (ML) classifiers before Deep Learning (DL) recently came into the spotlight. Hand-crafted descriptors are features extracted from images using a manually-defined algorithm based on expert knowledge [7]. Notable examples of hand-crafted descriptors algorithms are Haar-like features, Local Binary Pattern, Scale Invariant Feature Transform, Histogram of oriented Gradients. Popular ML classifiers include k-Nearest Neighbor, Decision Trees, Support Vector Machine (SVM), and Cascade Classifiers [8]–[15]. The training of ML parameters is conducted by using extracted descriptors and pre-defined labels. Depending on the descriptor, the performance of the classification is not consistent. It is necessary to determine which descriptor and classifier are more proper for each purpose. The more detection, tracking, and recognition modules the traditional method has, the more descriptors and classifiers are required to optimize performance of the system [16].

The ML-based approach is up to trial, error, and the engineer's judgment to determine proper descriptors and classifiers. On the other hand, the DL-based approach discovers the underlying patterns and descriptors that attempt to describe. The DL network also can work as a classifier in the process of training the DL model [16]. However, DL requires a large dataset to be representative and a sufficiently large model to describe the data. Early DL area was stuck in a proverbial dark age for years due to the vanishing gradient problem, the need for massive computing resources, and the absence of large datasets. The victory of AlexNet on the ImageNet Large-Scale Visual Recognition Challenge 2012 (ILSVRC 2012) [17] was a trigger for inspiring research to alleviate these problems. DL began to get attention in Computer Vision and has outstanding achievements in various fields, including art, music, games, and natural language. With this trend, DL has become mainstream in the automotive industry, where ML-based technology had been dominant.

Moreover, researches using images and DL based on laptops and embedded boards equipped with GPU have become more active because of the popularization of mobile GPUs [18], [26], [31]. Outstanding examples among the studies using mobile GPU are the Driver Drowsiness Detection Network (DDDN) based drowsiness detection [26] and Gaussian Mixture Model (GMM) based distraction detection algorithms [18]. DDDN-based drowsiness detection algorithm combines Convolutional Neural Networks (CNN) of two streams for eye and mouth for drowsiness detection. CNN of each stream has the advantage of being able to focus on areas that correspond to the eye or mouth. The architecture of DDDN enables real-time processing at the Jetson TK1. It also shows 93.84% drowsiness detection accuracy despite the use of uncomplicated CNN. GMM-based distraction detection algorithm can recognize the subdivided activities of a driver by using a segmented image based on GMM, even though they used a single CNN model. The detection accuracy of distraction caused by phone calls and text messages while driving is 93.2% and 94.5%, respectively. Also, driver distraction recognition can be processed at real-time over ten frames per second (FPS) on mobile GPU.

Despite these studies of conspicuous performance, it is necessary to study the driver status monitoring system on a light-weight SBC because the devices used on previous works are still expensive in commercialization. The work in [19] showed the potential for deploying driver status monitoring systems on light-weight SBCs. But, this simultaneously showed that real-time operation of the system is difficult on SBCs without GPUs. Compared to the high-end environment, the FPS of driver status recognition is significantly low on SBCs with limited computing resources. Because the accuracy of the system is dependent on real-time processed frames per second (PFPS), it is easier to lose the valuable instantly-occurring facial behavior of the driver when the time interval of the driver's facial behavior recognition increases. Eventually, this leads to a decrease in the accuracy of the driver's status recognition.

This work proposes an Adaptive Batch-Image based Driver Status Monitoring (ABI-DSM) system to alleviate this problem. The proposed system consists of Batch-Image Generation (BIG) and Batch-Image Processing (BIP) blocks. The BIP block has Facial Behavior Recognition (FBR) and Driver Status Recognition (DSR) modules. Using Batch-Image (BI), which is a batch of images accumulated between image processing intervals, enables the FBR module to get the information that can be missed during real-time processing. DSR module adaptively updates time series data using RBF module results of various sizes depending on the computing environment.

Although the Batch-Image helps to minimize the loss of information, processing Batch-Image that has more images occupies more memory. Memory size is one of the major factors in price competitiveness for commercialization. For example, if the memory of the Raspberry Pi and Jetson Nano doubles, their price is about 1.5 times higher. Then, it is necessary to reduce the amount of used memory without accuracy decrease. For the reduction of memory occupied by DL models, the design of ABI-DSM considers the following: First, the driver's facial behavior is used as the input of the DSR module to minimize the model size of the Long Short-Term Memory (LSTM) module for recognizing the driver's status [21]. Next, the FBR module recognizes the driver's facial behavior from the image without using face detection and tracking. Finally, the FBR module uses PydMobileNet [22], which is an advanced version of MobileNet models [23], [24] with stable performance in terms of accuracy and real-time processing, to process batch images on resource-restricted SBC.

The contributions of this paper can be summarized as follows.

- A novel struct system is applied to driver status recognition. Unlike existing studies that require re-training the

recognition model depending on the computing environment, the proposed system can recognize the driver status on a variety of materials without loss of the valuable information.
- PydMobileNet, proposed in our previous work, is applied to recognize driver facial behaviors. Experiments have demonstrated that the ABI-DSM system can be implemented without accuracy decrease by using PydMobileNet, which has a relatively small number of parameters compared to MobileNet, and that PydMobileNet is more efficient in SBC with lower specifications.

Section II describes related works and the motivation for this study. Details of the proposed ABI-DSM system, PydMobileNet, and LSTM models are in Section III. Section IV describes and discusses the experiments.

II. MOTIVATION

A. THE STRUCTURE OF DRIVER STATUS MONITORING SYSTEM

The types of vision-based driver status monitoring systems listed in Table 1 are as follows: First, most systems use the detection of parts of the face, including the eyes and mouth, and extract features from the patch image of the detected face parts. ML or DL classifiers recognize the driver status from the extracted features [26], [27], [30], [32]–[35]. The obtainable examples are eye closures and yawns. There are many open datasets of face parts, and it is relatively easy to collect images of face parts through web surfing. However, generally requiring individual feature extractors and detectors is a drawback, as this approach has limited accuracy and is vulnerable to occlusion.

TABLE 1. Related Works on the driver monitoring system.

	Detection	Detection		Used Library or ML/DL Models	Environment		FPS
		Face	Land-mark		CPU	GPU	
[25]	Drowsy			AlexNet, VGGNet, FlowNet	-*	-	-
[26]	Drowsy	✓	✓	MTCNN, DDDN	Jetson TX1	Jetson TX1	14.9
[27]	Distraction	✓	✓	Dlib, VGGNet	H**	H	89.2
[28]	Distraction	✓		MTDNN	H	H	33.0
[29]	Drowsy	✓	✓	MCT Adaboost, LBF Regression	H	H	-
[30]	Drowsy	✓	✓	Dlib, CNN, Viola-Jones Method	H	H	63.7
[31]	Distraction			GoogLeNet	Jetson TX1	Jetson TX1	11
[32]	Drowsy	✓	✓	Dlib, MTDMN	H	H	-
[33]	Fatigue	✓	✓	MTCNN, CNN	H	H	-
[18]	Distraction			AlexNet, GoogLeNet, ResNet	H	L***	14
[34]	Fatigue	✓	✓	MTCNN, ESR-Net, MSR-Net	H	H	-
[35]	Drowsy	✓	✓	Dlib, FHOG, MTCNN, SqueezeNet, CNN	H	-	17
[36]	Fatigue	✓	✓	Dlib, SVM, LR, DT, NB	-	-	-
[37]	Drowsy	✓	✓	SVM, DCCNN	H	H	20.1
[38]	Fatigue	✓	✓	Dlib, MTCNN, SVM, ANN	-	-	-
[39]	Fatigue	✓	✓	Dlib, Customed YOLOv3-tiny	H	H	24.3

* -: Not specified
** H: High performance
*** L: Low performance

Second, most other systems use facial landmarks extracted from the face image and the geometric information of the lines connecting facial landmarks [28], [29], [36]–[39]. This approach has the advantage of obtaining eye closing and mouth opening information by regression analysis from geometric features, unlike the former model that uses feature extraction and classification. Additional examples of features are the percentage closure of eyes, eye aspect ratio, mouth aspect ratio, eye feature vector, mouth feature vector, and eye closing duration. However, these geometric features depend on the precision of the positioning of landmarks and this is a challenging task that is still studied actively.

Approaches using a chain of face, eyes, mouth, and facial landmark detections have the potential to transmit errors from the former module to the following modules. For the ML-based approaches that have lower accuracy than DL-based approaches, potential errors have a more significant impact on the chain of detections based on ML. Implementing DL-based detection chains in the vehicle is typically difficult because of limited computing resources. To resolve these problems, a few researchers proposed methods not using detections of facial parts, such as the face, eye, mouth, and facial landmarks. By adding a multi-task layer to the CNN base, these methods obtain facial behaviors, such as head orientation, eye closing, and mouth opening at once. These obtain the results of facial behavior recognition that is not dependent on the detection of face, eyes, and mouth. Also, improving the processing performance is possible by reducing the latency caused by detecting face, eyes, and mouth [18], [19], [25].

However, these previous works have the following constraints. These systems, generally depending on high-performance GPUs, do not work on a SBC in real-time. The accuracy of driver status recognition becomes lower as the processing latency increases. Additionally, adaptively applying a trained model that recognizes the driver status on devices of different specifications is challenging. The batch-Image generated from BIG modules allows the ABI-DSM system to operate adaptively on a variety of devices that have different processing latency.

B. PydMobileNet FOR RECOGNIZING FACIAL BEHAVIORS

Since AlexNet won ILSVRC 2012, a variety of models, such as VGGNet [40] and GoogLeNet [41], have been introduced. After these newer models outperformed their predecessors in the following ILSVRC 2014, designing networks deeper and wider has become a trend to acquire superior performance.

VGGNet is a deep version of AlexNet, and GoogLeNet is a deep and wide model.

- VGGNet is an advanced version of AlexNet and has a structure like AlexNet in terms of sequentially using convolution, pooling, a fully-connected layer, and a softmax layer. This model uses factorized convolution with a stack of 3×3 filters instead of large-size filters, such as 5×5 and 7×7 filters. Consequently, VGGNet outperforms AlexNet by designing its structure in an uncomplicated manner, like AlexNet, but deeper.
- On the other hand, GoogLeNet has the Inception module instead of the fundamental convolution layer. The Inception module is designed with a combination of different kernels and has fewer parameters, considering the intricate structure, because it uses primary and spatial factorized convolution. The spatial factorized convolution is another alternative to the 3×3 convolution. Unlike the factorized convolution previously mentioned, spatial factorized convolution is a stack of 3×1 and 1×3 convolutions. Due to many other factors, GoogLeNet is complicated but has a lightweight structure 12 and 36 times smaller than AlexNet and VGGNet, respectively.

To cope with the resource restrictions in the FPGA and SBCs, compressing and optimizing a model has become another trend. SqueezeNet [42], Xception [43], and MobileNet [23], [24] are notable models. Their model has a stack of compressed module layers. SqueezeNet and Xception are built from the base of existing models such as AlexNet and GoogLeNet. MobileNet is an optimized model across both the model structure and module layers comprising the model.

- The design base of SqueezeNet is the structure of AlexNet, like VGGNet. The Fire module of SqueezeNet is a substitution of the convolution layer of AlexNet, and it consists of a squeeze convolution layer and an expanded layer. The squeeze convolution layer performs the function of reducing the channel of inputs. The expanded layer is like the residual convolution layer. Applying the Fire module reduces the computational cost consumed by the primary 3×3 convolution. Consequently, SqueezeNet has 50 times fewer parameters than AlexNet and can occupy less memory, making it possible to deploy it on a FPGA.
- Xception is a lightweight version of the Inception model, made smaller by using the depthwise separable convolution (DWConv). DWConv was primarily introduced in [44], but it was applied only to parts of the overall model. Xception uses the depthwise separable filter scaled up over the model, named DWConv. By using just an extreme version of the Inception module instead of the primary Inception module, Xception built from the base of InceptionV1 has a number of parameters similar to InceptionV3. Furthermore, the accuracy of Xception is greater than that of InceptionV3.
- MobileNet is the designed model by optimizing both macro and micro components. In the micro realm, MobileNetV1 uses the DWConv instead of the primary convolution. MobileNetV2 substitutes the DWConv of MobileNetV1 with an Inverted Residual module. The Inverted Residual module is a deformed Residual module by varying the channel depth values of the in/output layer and hidden layers. In the Inverted Residual module, the first layer with a 1×1 convolution reduces the number of input channels to the 3×3 depthwise convolution layer that consumes the most computation time. It similarly works to the squeeze convolution layer of SqueezeNet. In the macro respect, the MobileNet does not use a pooling layer except for the last layers. Instead, it uses a stride of depthwise convolution to reduce the width and height of the feature maps. MobileNetV2 has fewer parameters than the ShuffleNet and NasNet-A. MobileNetV2 also requires fewer operations (FLOPs) than the SqueezeNet, and it outperforms SqueezeNet.

Inception, Xception, SqueezeNet, and MobileNet have stacks of the Inception, Xception, Fire, Inverted Residual modules, respectively. The commonalities of models implemented by stacks of modules since AlexNet are summarized as follows: First, decrease the input dimension of the expensive 3×3 and 5×5 convolutions by the 1×1 convolution. It is a bottleneck and has the effect of reducing the computation cost of the 3×3 and 5×5 convolution layers. The first 1×1 convolution of Inception, Xception, and SqueezeNet modules work as the bottleneck. The squeeze convolution layer of the Fire module is also included.

Next, there is dimensional expansion after the 3×3 and 5×5 convolutions. The dimensional expansion generally generates a wider variety of features. Because repeated dimensional expansion results in an excessive increase in the computational cost, the dimensional expansion is not applied to every layer in the AlexNet and VGGNet. Alternatively, the expanded dimension is shrunken by a 1×1 reduction convolution of the same module or bottleneck of the following module. In the Inception and Xception modules, the dimension is expanded by concatenating the outputs of 3×3 or 5×5 convolutions. The Expansion and Expand layers respectively expand the dimensions of the Inverted Residual and Fire modules.

Lastly, those modules include a residual layer. Although each module has different structures, such as a 1×1 convolution or bypass connection, they perform the same role as an Identity Mapping proposed in the ResNet. Identity Mapping maintains information that is likely to be lost by bringing features from the previous layer to the next layer.

Among the characteristics of InceptionNet and MobileNetV2, motivations of PydMobileNet are as follows.InceptionNet uses convolutions of different sizes in the same module layer. Using convolutions of different-size kernels at the same time in a module allows the module to obtain more varied spatial information from the input layer. The relatively high accuracy of the Inception-like models proves it. MobileNetV2 has an uncomplicated structure. It is sufficiently lightweight to be used in mobile devices.

To process Batch-Image in real-time and achieve reliable performance on a SBC, the design of PydMobileNet takes into account both the commonalities and distinguishing above. The base structure of PydMobileNet is the same as that of MobileNetV2. The PydResidual module substitutes the inverted Residual module, and this module has 3×3, 5×5, and 7×7 depthwise separable convolutions in parallel. The details of PydMobileNet are presented in Section III.

III. PROPOSED DRIVER STATUS MONITORING SYSTEM

The Adaptive Batch-Image based Driver Status Monitoring (ABI-DSM) system has Batch-Image Generation (BIG) and Batch-Image Processing (BIP) blocks, as shown in Figure 1. Through socket communication, the BIG block sends a batch image whenever the BIP block requests one. While the BIP block recognizes the driver status, the BIG block generates a batch image (BI). Each block operates on a different thread.

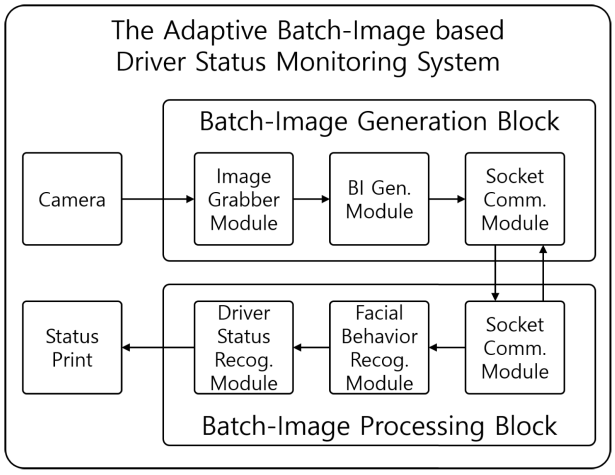


FIGURE 1. The Adaptive Batch-Image based Driver Status Monitoring.

The hardware specification of the ABI-DSM system is NVIDIA Jetson Nano with Raspberry Pi NoIR Camera V2.0 and 850nm wavelength infra-red led light. The eye blink duration is about 239.3(±20.2) millisecond, and facial behavior needs to be recognized more than twice during this duration [19], [20]. At least 9.13 FPS performance is required for driver status recognition. NVIDIA Jetson Nano board (Jetson Nano) satisfies this prerequisite. Jetson Nano is a GPU-equipped SBC, and its price is only about twice that of a Raspberry Pi 4 with the same memory size. Raspberry Pi NoIR Camera is connected to NVIDIA Jetson Nano via Camera Serial Interface (CSI) with a low load of CPU and memory rather than a USB interface. Because it has no IR filter, it is possible to capture images at night with 850nm wavelength infra-red light.

A. BATCH-IMAGE GENERATION BLOCK

The BIG block consists of Image Grabber (IG), Batch-Image Generation (BG), and Socket Communication (SC) modules. The IG module of BIG on the Jetson Nano captures twenty images per second from a camera. Whenever the IG module captures the image, it sends this to the BG module. The BG module appends the received image from the IG module to the batch image (BI). Each SC module in the BIG and BIP blocks has two ports. Via one port, the Request and Acknowledge signals are sent. Via the other port, the BIG block sends the batch image to the BIP block.

B. BATCH-IMAGE PROCESSING BLOCK

The BIP block consists of Facial Behavior Recognition (FBR), Driver Status Recognition (DSR), and SC modules. The SC module of the BIP block requests the batch image to that of the BIG block and transfers the received batch image to the FBR module. The FBR module recognizes the facial behavior of the batch size from the batch image. The DSR module receives the facial behaviors batch from the FBR module and recognizes the driver's status. To better solve the problems mentioned in Section I, we design the BIP block in consideration of several goals. The first goal is to optimize the CNN model recognizing facial behaviors from the batch image. The next goal is to reduce the number of fully connected nodes that demand more memory. Finally, each model is trained by Multi-Task Learning (MTL), such that FBR and DSR modules conduct multiple tasks with each model base. The following describes details of the FBR and DBR modules.

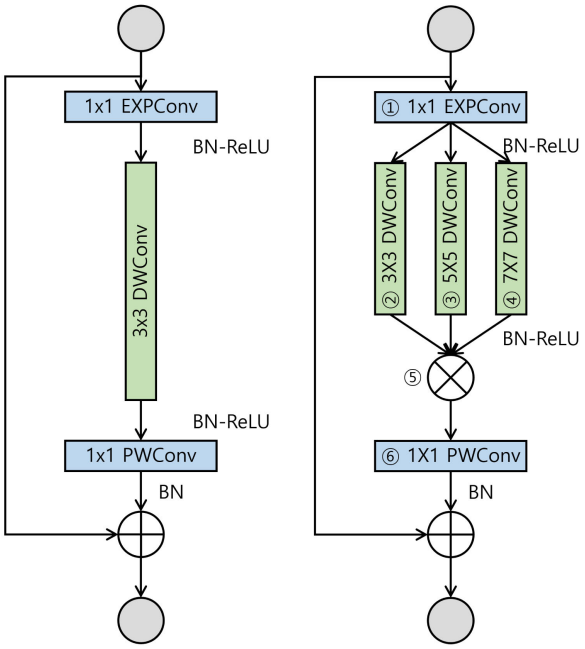
C. FACIAL BEHAVIOR RECOGNITION MODULE

Although Jetson Nano has GPU cores, a set of batch images is a burden for computation using a SBC. For batch image processing under restricted computing resources, reducing the parameters of a model is helpful. Accordingly, the FBR module uses PydMobileNet. PydMobileNet is the version that uses the PydResidual layer as a substitute for the Inverted Residual layer of MobileNetV2. PydMobileNet has fewer parameters than MobileNetV2, with similar accuracy.

1) RESIDUAL LAYER

PydMobileNet is based on the architecture of MobileNetV2 and it consists of a stack of PydResidual modules. MobileNet-like models are generally less accurate than Inception-like models. As with MobileNet-like models, Inception-like models consist of a stack of several different modules. The distinction is that modules in the Inception-like model use a combination of convolutions with different kernel sizes.

Inspired by this, we designed the PydResidual module to contain several Depthwise Convolution (DWConv) layers with different-size kernels instead of using one size, as with MobileNetV2, as shown in Figure 2. Table 2 describes the shapes of the input, output, and kernel layers in the PydRedisual module. The DWConv layers of the PydResidual module share the output of the Depth Expand layer. The outputs of the DWConv layers are concatenated and become the input of the Pointwise Convolution (PWConv) layer.



(a) Inverted Residual module layer (b) PydResidual module layer

FIGURE 2. The Structural Comparison between Inverted Residual and PydResidual Module Layers.

TABLE 2. Details of PydResidual layer.

Layer	Input	kernel shape	Output	Layer type
1	$h \times w \times d_i$	$1^2 \times d_i \times d_o{}'$	$h \times w \times d_o{}'$	EXPConv
2	$h \times w \times d_o{}'$	$3^2 \times d_o{}'$	$h \times w \times d_o{}'$	DWConv
3	$h \times w \times d_o{}'$	$5^2 \times d_o{}'$	$h \times w \times d_o{}'$	DWConv
4	$h \times w \times d_o{}'$	$7^2 \times d_o{}'$	$h \times w \times d_o{}'$	DWConv
5	$[h \times w \times d_o{}', h \times w \times d_o{}']$	-	$h \times w \times d_o{}''$	Concatenation
6	$h \times w \times d_o{}''$	$1^2 \times d_o{}'' \times d_o$	$h \times w \times d_o$	PWConv

By using several DWConv layers in the PydResidual module, the input of the PWConv layer is N times that of the DWConv layer. N is the number of DWConv layers. To eliminate repetitive expansion, the Depth Expand layer of the PydResidual module expands the input depth (d_i) of the PydResidual module layer by $\frac{t}{N}$ instead of the expanding factor t in the Inverted Residual module. The input depth ($d_o{}'$) of each DWConv layer in the PydResidual module is N times smaller than that of the DWConv layer in the Inverted Residual module. As a result, the concatenated output depth ($d_o{}''$) of the DWConv layers is equal to the output depth of the DWConv layer in the Inverted Residual module. The output depth (d_o) of PWConv is double or equivalent to (d_i), as with MobileNetV2. The output depth (d_o) is usually less than ($d_o{}''$).

The equations calculating the number of parameters in the Expand, DWConv, and PWConv layers are Eqs. (1), (2), (3), and (4), respectively. The Expand layer includes 1 × 1 PWConv, Batch Normalization (BN), and ReLU Activation (ReLU) layers. The number of 1 × 1 PWConv layer parameters is the product of the depth values of input (d_i) and output ($d_o{}'$) features maps. The number of BN and ReLU layers parameters is $d_o{}'$, respectively. The the Expand layer parameters (P_E) decrease according to the number (N) of DWConv layers. The number of the DWConv layer parameters (P_D) with $k_n \times k_n$ filter is the product of kernel's parameters ($k_n{}^2$) and $d_o{}'$. It changes according to the size of the kernel as well as N. But, the number of PWConv layer parameters (P_P) does not change. P_{IR} and P_{PR} of Eqs. (5) and (6) are the total number of parameters on Inverted Residual and PydResidual module layers, respectively.

$$d_o' = \frac{d_o''}{N} = \frac{t \times d_i}{N} \tag{1}$$

$$P_E(d_i, d_o{}', N) = d_i \times d_o{}' + 2d_o{}' \tag{2}$$

$$P_D(d_o{}', k_n, N) = k_n{}^2 \times d_o{}' + 2d_o{}' \tag{3}$$

$$P_P(d_o{}'', d_o) = d_o{}'' \times d_o + 2d_o \tag{4}$$

$$P_{IR} = P_E(d_i, d_o{}', 1) + P_D(d_o{}', k_1, 1) + P_P(d_o{}'', d_o) \tag{5}$$

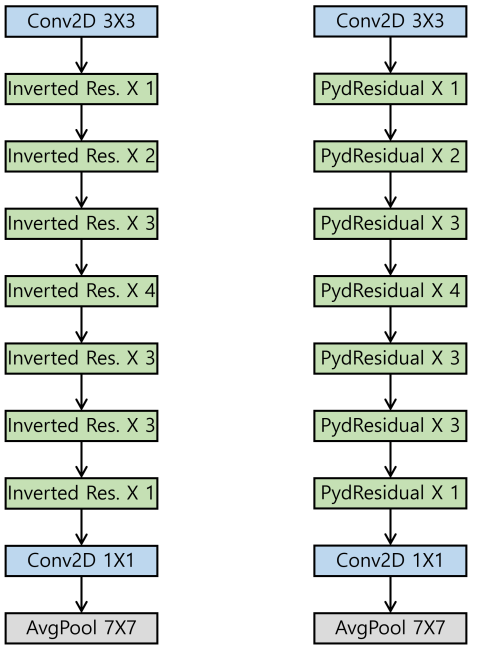
$$P_{PR} = P_E(d_i, d_o{}', N) + \sum_{n}^{N} P_D(d_o{}', k_n, N) + P_P(d_o{}'', d_o) \tag{6}$$

2) PydMobileNet BASE

PydMobileNet extracts feature for recognizing behaviors from the driver image. PydMobileNet is an improved version of MobileNetV2 that has a simple architecture that enables real-time use. MobileNetV2 consists of a stack of Inverted Residual modules. By using Depthwise Separable Convolutions in the module, MobileNet-like models have fewer parameters than other models. Hence, MobileNetV2 can work in a real-time mobile environment. Similarly, PydMobileNet not only has fewer parameters but also enables real-time operations in the restricted environment.

Figure 3 shows a structural comparison between PydMobileNet and MobileNetV2. In Table 3, the input/output shape, expanding factor (t), the number of output channels (c), iteration of the module layer (n), and stride of DWConv (s) on PydMobileNet are equal to those of MobileNetV2. However, the PydResidual module replaces the Inverted Residual module in the MobileV2 model. Although the parameters of the PydResidual module by using N DWConv layers increases, the use of larger kernel convolutions enables the utilization of more spatial information.

Table 4 lists the number of parameters in the Expand, DWConv, and PWConv layers for each module in MobileNetV2. For reducing the number of total parameters, the module uses the Expand layer in MobileNetV2. However, Table 4 shows parameters in the Expand layer account for a quarter of the total number of parameters. Compared to Table 4, Table 5 shows a slight increase in the parameters of the DWConv Layer. But, the parameters of the Expand layer significantly decrease. Consequently, the parameters



(a) MobileNetV2 base (b) PydMobileNet base

FIGURE 3. The Structural Comparison between MobileNetV2 and PydMobileNet models.

TABLE 3. Details of PydMobileNet base.

Input	Operator	Output	t	c	n	s
$224^2 \times 3$	Conv2D 3×3	$112^2 \times 32$	-	32	1	2
$112^2 \times 32$	PydResidual layer	$112^2 \times 16$	3	16	1	1
$112^2 \times 16$	PydResidual layer	$56^2 \times 24$	6	24	2	2
$56^2 \times 24$	PydResidual layer	$28^2 \times 32$	6	32	3	2
$28^2 \times 32$	PydResidual layer	$14^2 \times 64$	6	64	4	2
$14^2 \times 64$	PydResidual layer	$14^2 \times 96$	6	96	3	1
$14^2 \times 96$	PydResidual layer	$7^2 \times 160$	6	160	3	2
$7^2 \times 160$	PydResidual layer	$7^2 \times 320$	6	320	1	1
$7^2 \times 320$	Conv2D 1×1	$7^2 \times 1280$	-	1280	1	1
$7^2 \times 1280$	AvgPool 7×7	1×1280	-	-	1	-

of PydMobileNet are fewer than those of MobileNet. The number of parameters in Table 5 can be calculated by Eqs. (2), (3), and (4). This reduction of parameters proves that sharing the role of the Expand layer with N DWConv layers and the Depth Expansion by $\frac{t}{N}$ can reduce the burden of expanding the depth. The number of parameters in Table 4 and 5 can be calculated by Eqs. (2), (3), and (4).

In the Eq. (7), R is ratio between the number of parameters on Inverted Residual and PydResidual module layers. It is possible to compare the total number of parameters between those module layers. R' of Eq. (8) that excepts terms not required for comparison in Eq. (7) can substitute R. The PydResidual module layer contains DWConv layers with $3 \times 3, 5 \times 5, 7 \times 7$ kernels in this paper. For these sizes of kernels, N is three and $k_n = \{3, 5, 7\}$. Given these prerequisites, Eq. (8) is summarized into Eq. (9). According to Eq. (9), the number of parameters in the PydResidual module layer is less than those of the Inverted Residual module layer if d_i is above 26.

TABLE 4. Parameters on MobileNetV2 base.

Layers	Expand	DWConv	PWConv	Total
1	-	-	-	928
2	-	352	544	896
3	1,728	1,056	2,352	5,136
4	3,744	1,584	3,504	8,832
5	3,744	1,584	4,672	10,000
6,7	6,528	2,112	6,208	14,848
8	6,528	2,112	12,416	21,056
9,10,11	25,344	4,224	24,704	54,272
12	25,344	4,224	37,056	66,624
13,14	56,448	6,336	55,488	118,272
15	56,448	6,336	92,480	155,264
16,17	155,520	10,560	153,920	320,000
18	155,520	10,560	307,840	473,920
19	-	-	-	412,160
Total	-	-	-	2,223,872

TABLE 5. Parameters on PydMobileNet base.

Layers	Expand	DWConv			PWConv	Total
		3×3	5×5	7×7		
1	-	-	-	-	-	928
2	-	352	864	1,632	1,568	4,416
3	576	352	864	1,632	2,352	5,776
4	1,248	528	1,296	2,448	3,504	9,024
5	1,248	528	1,296	2,448	4,672	10,192
6,7	2,176	704	1,728	3,264	6,208	14,080
8	2,176	704	1,728	3,264	12,416	20,288
9,10,11	8,448	1,408	3,456	6,528	24,704	44,544
12	8,448	1,408	3,456	6,528	37,056	56,896
13,14	18,816	2,112	5,184	9,792	55,488	91,392
15	18,816	2,112	5,184	9,792	92,480	128,384
16,17	51,840	3,520	8,640	16,320	153,920	234,240
18	51,840	3,520	8,640	16,320	307,840	388,160
19	-	-	-	-	-	412,160
Total	-	-	-	-	-	1,849,280

According to Tables 4 and 5, the parameters of PydMobileNet are fewer than those of MobileNetV2 after the input depth becomes 32. The number of parameters in PydMobileNet is about 83.16% of the number of parameters in MobileNetV2.

$$R = \frac{P_{IR}}{P_{PR}} = \frac{P_E(d_i, d_o', 1) + P_D(d_o', k_1, 1) + P_P(d_o'', d_o)}{P_E(d_i, d_o', N) + \sum_{n=1}^{N} P_D(d_o', k_n, N) + P_P(d_o'', d_o)} \tag{7}$$

$$\begin{aligned} R' &= \frac{P_E(d_i, d_o', 1) + P_D(d_o', k_1, 1)}{P_E(d_i, d_o', N) + \sum_{n=1}^{N} P_D(d_o', k_n, N)} \\ &= \frac{d_i \times d_o' + 2d_o' + {k_1}^2 \times d_o' + 2d_o'}{d_i \times \frac{d_o'}{N} + 2\frac{d_o'}{N} + \sum_{n=1}^{N}({k_n}^2 \times \frac{d_o'}{N} + 2\frac{d_o'}{N})} \\ &= \frac{d_i + {k_1}^2 + 4}{\frac{d_i}{N} + 2\frac{1}{N} + \sum_{n=1}^{N}({k_n}^2 \times \frac{1}{N} + 2\frac{1}{N})} \\ &= \frac{d_i + {k_1}^2 + 4}{\frac{1}{N}(d_i + 2N + 2 + \sum_{n=1}^{N}({k_n}^2))} \end{aligned} \tag{8}$$

$$R' = \frac{3d_i + 39}{d_i + 91} \tag{9}$$

3) MULTI-TASK LEARNING

Primarily, independent models are trained for each task to implement multiple tasks. In this case, the overfitting is likely

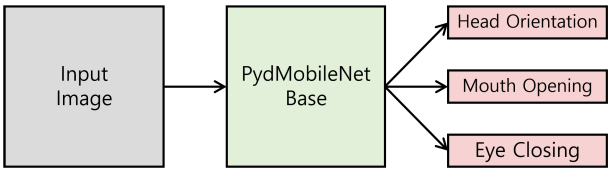


FIGURE 4. Multi-task PydMobileNet.

to occur. On the other hand, Multi-Task Learning (MTL) enables the model to avoid overfitting by sharing the model base. Each task works as auxiliary layer for other tasks. The auxiliary layer generally improve the performance. Also, a single model simultaneously performs multiple tasks from a single input data. The MTCNN model is an example model of using MTL. This model simultaneously detects a face and facial landmarks in the image.

In the Facial Behavior Recognition module, Multi-Task (MT) layer recognizes the driver's facial behaviors from the output vector of the PydMobileNet base. Facial behaviors include head orientation (HO), mouth opening (MO), and eye closing (EC). The categorization of HO includes front, up, down, left, and right. The MO and EC parameters are a binary classification for close and open. Figure 4 is the diagram of Multi-Task PydMobileNet (MT-PydMobileNet) which is PydMobileNet trained by MTL.

D. DRIVER STATUS RECOGNITION

1) MULTI-TASK LONG SHORT-TERM MEMORY

The Recurrent Neural Network model (RNN) is applicable in a variety of fields such as speech recognition, language translation, and human action recognition. RNN models can extract information that changes along a temporal sequence from time-series data. Especially, the Long Short-Term Memory model (LSTM), one of the RNN models, is a model developed to deal with the vanishing point problem of common RNN by using regulators. LSTM is trained by MTL (MT-LSTM) and able to simultaneously recognize three statuses, including distraction, fatigue, and drowsiness.

Figure 5 shows a Vanilla LSTM model that consists of an LSTM cell with a single hidden layer. The LSTM cell consists

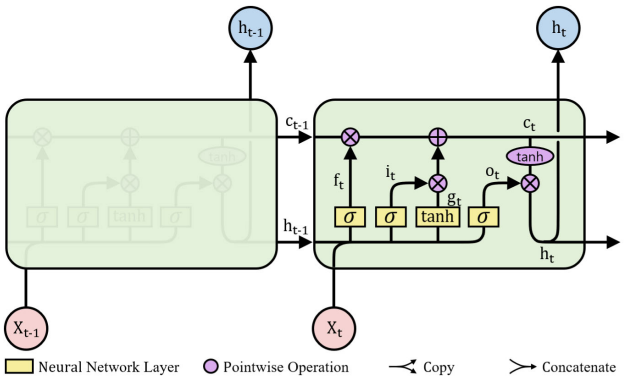


FIGURE 5. Vanilla Long Short-Term Memory Structure.

of the input(i_t), forget(f_t), cell($\tilde{c}_t$), and output(o_t) gates. The LSTM network performs calculations with the input sequence $x = (x_1, \ldots, x_t)$ to produce outputs $h = (h_1, \ldots, h_t)$ at time t by using the several gates and states as follows: In Eq. (10), the c_t is the activated proportionate sum of the previous cell state c_{t-1} and cell gate $\tilde{c}_t$ according to f_t and i_t. LSTM outputs the hidden state h_t which is the element-wise product ($\odot$) of the vectors, which are activated c_t by tanh and o_t.

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ \tilde{c}_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{10}$$

As shown in Figure 6, MT-LSTM has a many-to-one type structure that recognizes the driver status along the temporal sequence of the driver's facial behaviors over a recent time period Tp. LSTM base is two stacked hidden layers, and each hidden layer has an LSTM cell. Feedforwarding of the stacked hidden layers repeats as many times as the length of the input sequence. The fully-connected (FC) layer and three binary classification layers follow the LSTM base. The binary classification layer includes the FC layer and softmax.

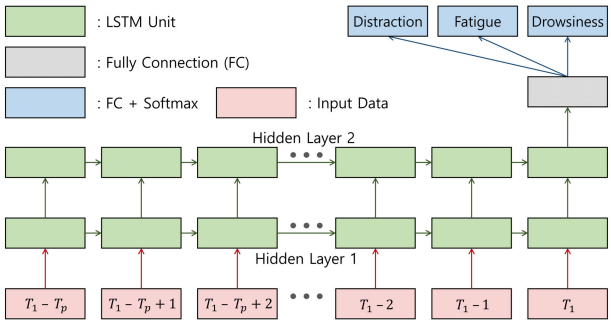


FIGURE 6. MT-LSTM Structure.

2) TRAINING MT-LSTM

Most research using both CNN and RNN models have used a flattened output of the CNN base for the input to the RNN model. Although different for each target application, the length of the CNN base output is generally around a thousand. Because RNN models consist of fully-connected (FC) layers, parameters in the FC layer hugely increase and decrease according to the input and output size of the FC layer. Furthermore, LSTM uses more FC layers due to the regulation gates. To reduce the number of parameters for real-time operation in the restricted environment, it is essential to decrease the input and output sizes of the FC layers.

The number of parameters on the MT-LSTM model can be calculated by Eq. (11). This depends on the length of the

TABLE 6. Parameters on MT-LSTM.

Layer	Params	input=3 hidden=4^3		input=1280 hidden=4^3		input=1280 hidden=2×4^3	
		W	b	W	b	W	b
Hidden	W_{ih}, b_{ih}	768	256	327,680	256	655,360	512
Layer 1	W_{hh}, b_{hh}	16,384	256	16,384	256	65,536	512
Hidden	W_{ih}, b_{ih}	16,384	256	16,384	256	65,536	512
Layer 2	W_{hh}, b_{hh}	16,384	256	16,384	256	65,536	512
FC	-	4,096	64	4,096	64	16,384	128
CLS 1	-	128	2	128	2	256	2
CLS 2	-	128	2	128	2	256	2
CLS 3	-	128	2	128	2	256	2
total	-	55,494		382,406		871,302	

LSTM cell input (N_i) and the size of the hidden layer (N_h). Table 6 shows the comparison of parameters when the size of the input and hidden increases. When using the output of the CNN base instead of the driver's facial behaviors vector, MT-LSTM needs as many parameters as the number of parameters that decrease by using PydMobileNet. Increasing the input and hidden sizes of the LSTM cell has the result that the number of parameters in MT-LSTM is as many as half parameters in the PydMobileNet base. The LSTM cell uses the facial behaviors vector as an input to decrease the number of parameters. In this paper, the number of classifications (C_n), N_i, and N_h are three, three, and 4^{C_n}, respectively.

$$\begin{aligned}(W_{ih}, b_{ih}) &= \{(W_{ik}, b_{ik}) \mid k = i, f, g, o\} \\ (W_{hh}, b_{hh}) &= \{(W_{hk}, b_{hk}) \mid k = i, f, g, o\} \\ P_{(W_{ih}, b_{ih})} &= 4 \times N_i \times N_h + 4 \times N_h \\ P_{(W_{hh}, b_{hh})} &= 4 \times N_h \times N_h + 4 \times N_h \end{aligned} \tag{11}$$

3) GENERATING A TIME SERIES OF DRIVER's FACIAL BEHAVIORS

Computing resources vary depending on the operating environment of the system. For MobileNets, GPU-equipped Desktops can run the model at over 30 frames per second (FPS). On the other hand, SBCs equipped with only CPUs barely handle one or two frames per second. The driver's status recognition is quite sensitive to the time factor. Not considering the time domain, using sequential data result in a fault result.

For example, the system is supposed to operate a system with a model trained from 20FPS data on an SBC equipped with only CPUs. If four consecutive eye closures are detected, the system regards that the eyes were closed for two seconds and should give a warning. From the viewpoint of the model trained by 20FPS data, eyes are closed for 0.2 seconds. The system will recognize the driver's status as a general eye blink. Therefore, the system needs to include a block generating time series data of driver's behaviors for considering the characteristic of the time domain.

Training the driver's status recognition model to suit the operating environment is an ideal method. But, training the model regarding every situation is impossible. Even if numerous models are trained, the deviation of the instance processing latency is significant even when operating the system in the same environment. It is tricky to pick the proper model for every recognition. Therefore, the DSR module generates a queue of time-series facial behaviors by considering processing time.

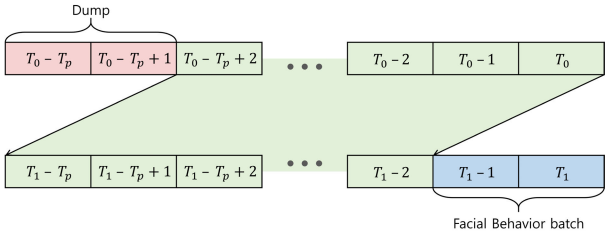


FIGURE 7. Generating Time-Series Dataset.

Figure 7 describes the procedure for generating the queue of time series facial behaviors. The BIG block generates a batch image while the BIP block recognizes the driver's status. The size of the batch image depends on each processing time of the BIP. The FBR module generates a facial behavior batch of the same size as the batch image. The DBR module dequeues data as much as the size of the facial behavior batch received from the FBR module, and enqueues the facial behavior batch. In this paper, the data has assigned labels of driver status based on a sequence of data of ten seconds. The acquisition rate of images is about 20FPS. Then, the size of the queue is two hundred.

IV. EXPERIMENT RESULTS

A. ENVIRONMENT

The system contains Raspberry pi camera version 2 without an IR filter. The camera is connected to an NVIDIA Jetson Nano with Camera Serial interface (CSI). The captured image has a resolution of 1280×720 pixels. An established infra-red led light with an $850nm$ wavelength wave has an illumination sensor and brightens the driver's face at night. The camera module is built into a self-made case and mounted on the driver's front dashboard, as shown in Figure 8.

This work conducted experiments in a vehicle and driving simulator. The vehicle experiment is an experiment to collect training and test data for the driver's facial behavior recognition model. To obtain images by considering the natural light of various angles, the direction of head of the vehicle was changed to be one of the four cardinal points (north, south, east, and west) per day. For taking into account the varying amounts of illumination, the vehicle experiment was conducted during a clear day, cloudy day, and at night. Unlike images taken during the cloudless day, these taken at night do not have color information but have facial textures to recognize the driver's facial behavior.

In the driving simulator experiment, we collected a dataset to train and test the model recognizing the driver's status. Unlike the vehicle experiments, the driving simulator is indoors, and the infra-red led light not used. The map of the

(a) Vehicle Environment (b) Simulator Environment

FIGURE 8. Dataset Acquisition Environment.

TABLE 7. Participants.

Environments	Ages	20-29	30-39	40-49	50-	Total
Vehicle	Man	10	5	2	1	18
Vehicle	Woman	2	5	8	7	22
Simulator	Man	3	7	8	5	23
Simulator	Woman	4	4	5	4	17
Total		19	21	23	17	80

driving simulator is a combination of a virtual highway and an urban road modeled on Sangam, Korea. We instructed the subjects to imitate the driver's status according to the given criteria while driving. Criteria of our previous research are used. [19]

B. DATASETS

A total of eighty people from Ulsan, Korea were recruited for participants. They had in the same ratios of age and gender, as shown in Table 7. Thirty subjects participated in the simulator experiment to acquire the dataset for the driver status recognition module. Outdoor experiments for the cloudless day, cloudy day, and at night have a proportion of 2 : 1 : 1. Thirty subjects participated in the vehicle experiment to acquire the dataset for the facial behavior recognition module, and the others participated in both experiments. The data of ten subjects, who participated in only outdoor experiment, comprise test dataset. The training and validation dataset consists of the data of the rest of the subjects who participated in each experiment.

The labels of datasets to train and test the driver facial behavior model consist of five face orientations, two mouth opening states, and two eye-closing categories. There are twenty combinations of cases. For each subject, we sampled 470 images for each case. The number of total training images is 376,000, which are sampled from a total of about 865,000 images. The four-folds validation dataset for the training and validation model consists of data form the forty participants.

The label of datasets to train and test the driver's status model consist of binary classifications, including distraction, fatigue, and drowsiness. The combination of fatigue and drowsiness has three labels because drowsiness occurs after fatigue. Recognition of distractions is performed independently of drowsiness and fatigue. The total number of images acquired is about 412,000. The number of status data images is 411,800 because the units defining driver status is two hundred images. The number of four-fold validation dataset for forty participants is about 3,312,000, and that of the test dataset for ten is about 79,000.

C. EXPERIMENTS

For training and validating the Facial Behavior Recognition model, we use datasets of forty people divided into four groups of ten. Three groups are for training the model. The remaining group is for validation. The process was repeated to train and validate the model four times by changing the validation set. Table 8 shows the average accuracy of the 4-fold validation for three tasks that include Head Orientation(HO), Mouth Opening(MO), and Eye Closing(EC). To measure the performance, we use the Accuracy(A) of Eq. (12) that uses the number(N_x) of True Positive(TP), False Negative(FN), False Positive(FP), and True Negative(TN) cases. The accuracy of Eq. (12) is the ratio of the true results among the total datasets.

$$A = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FN} + N_{FP} + N_{TN}} \tag{12}$$

The training part uses the Adam Optimizer and different learning rates depending on the model. Table 8 lists the results of the epoch with the highest average Precision of the three tasks out of a total of 30 epochs. InceptionNet outperforms the other models for the training, and the difference between training and validation accuracy is the smallest. Subsequently, ResNet and MobileNet showed high accuracy in training and validation, respectively. Compared to training accuracy, the average accuracy of MobileNet and PydMobileNet on the validation decreases relatively less, but that of the other models decreases significantly.

TABLE 8. Accuracy of Facial Behaviors Recognition Module for Training and Validation Datasets.

CNN Models	Training Accuracy				Validation Accuracy			
	HO	MO	EC	Avg.	HO	MO	EC	Avg.
AlexNet	98.0	99.9	98.0	98.6	78.7	96.8	71.7	82.4
InceptionNet	99.4	100	99.9	**99.8**	83.9	99.4	95.2	92.8
ResNet	99.2	99.9	99.8	**99.7**	82.8	98.6	92.1	91.2
ShuffleNet	97.2	99.8	97.1	98.0	79.7	96.4	81.4	85.8
SqueezeNet	92.1	98.5	81.8	90.8	77.5	96.1	72.6	82.1
MobileNetV2	97.3	99.9	99.3	98.8	88.3	99.0	94.4	**93.9**
PydMobileNet	98.9	99.9	99.8	99.5	87.3	98.9	93.7	**93.3**

For testing the trained Facial Behavior Recognition module, the datasets of ten people are used. To measure the performance for testing datasets, the measures of Precision(P), Recall(R), and F-score(F) are used. As shown in Eq. (13), precision is the ratio of true positives among positive results of recognition. Recall of Eq. (14) is the ratio of true positives

TABLE 9. Precision, Recall, and F1-score of Facial Behaviors Recognition Module for Test Dataset.

CNN Models	Head Orientation(HO)			Mouth Opening(MO)			Eyes Closing(EC)			Total Avg.
	P	R	F	P	R	F	P	R	F	P
AlexNet	73.0	62.7	59.4	70.4	86.3	71.7	78.4	61.7	61.3	73.9
InceptionNet	85.2	90.4	86.4	85.2	91.7	87.6	**90.9**	89.5	90.0	87.1
ResNet	**87.4**	63.4	69.1	85.6	92.9	87.3	86.4	79.6	81.9	86.5
ShuffleNet	71.7	63.8	63.3	57.6	85.9	57.9	72.5	74.5	64.9	67.3
SqueezeNet	33.0	33.1	27.9	55.6	61.3	38.3	57.3	55.9	48.1	48.6
MobileNetV2	**85.6**	97.8	90.6	**96.9**	99.7	98.3	**87.2**	95.8	90.7	**89.9**
PydMobileNet	84.5	96.3	89.3	**97.6**	99.6	98.6	84.4	93.6	88.0	**88.8**

to the number of positives in the ground truth.The F-score of Eq. (15) is the harmonic mean of Precision and Recall. This paper uses the F1-score with a β of one. Table 9 shows the Precision, Recall, and F-score of three tasks run on the test dataset. MobileNetV2 outperforms the other models across three tasks, followed by PydMobileNet.

$$P = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{13}$$

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{14}$$

$$F = (1 + \beta^2) \times \frac{P \times R}{\beta^2 \times P + R} \tag{15}$$

The data of the 30 participants in the simulator experiment is for training the driver's state recognition model. The data of 10 participants is for the testing models. To measure the performance of the driver's distraction, fatigue, and drowsiness recognition, we also use Precision, Recall, and the F-score. As shown in Table 10, the Decision Tree model shows the highest Precision in distraction recognition, which is a relatively simple task. According to the average(avg.) Precision of three tasks, LSTM outperforms other models in three tasks.

For implementing the system, we use Scikit-learn for the Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). For Training the Multi-Layer Perceptron (MLP) and LSTM, we use PyTorch. The maximum depth of the Decision Tree and Random Forest classifiers is five. We use an SVM with a linear kernel (LinearSVM) for real-time use and trains LinearSVM with $C = \{10, 1, 0.1, 0.01, 0.001\}$. Considering the average precision for three tasks, we use 0.01 for C value. The used MLPs have one (MLP1) and two (MLP2) hidden layers. The size of the hidden layers is the same as that of LSTM. Training MLPs and LSTMs iterate at 30 epoch. The training error is saturated under the 25^{th} epoch.

Table 11 lists the number of parameters and FLOPs of the CNN bases and shows the FPS of the DSM system without the BIG block. SqueezeNet has the smallest number of parameters, and the FLOPs of ShuffleNet is lower than others. The number of parameters and FLOPs of PydMobileNet are relatively low compared to the others. Even though PydMobileNet improves both compared to MobileNetV2, the FPS

TABLE 10. Precision, Recall, and F1-score of Driver Status Recognition Module for Test Dataset.

Algorithms or Models	Distraction			Fatigue			Drowsiness			Avg.
	P	R	F	P	R	F	P	R	F	P
DT	**99.9**	99.9	99.9	93.7	90.2	91.6	**99.3**	99.4	99.3	**97.6**
RF	97.9	99.7	98.8	95.7	92.4	93.8	97.8	99.6	98.6	97.1
LinearSVM	87.8	88.4	88.1	97.5	95.4	96.3	92.9	89.8	91.3	92.7
MLP	86.1	87.0	86.5	96.9	96.9	96.9	88.7	88.8	88.7	90.5
MLP2	88.2	89.2	88.7	**97.4**	95.0	96.1	92.6	89.5	90.9	92.7
LSTM	**99.7**	99.9	99.8	**99.8**	99.8	99.8	**99.5**	99.8	99.7	**99.7**

TABLE 11. FLOPs and Parameters of CNN base and FLOPs of the Traditional DSM system according to CNN base.

CNN base	Parameters	FLOPs	FPS
AlexNet	61.100M	821.109M	**13.210**
InceptionNet	27.161M	5,771.588M	6.851
ResNet	25.557M	4,136.628M	8.744
ShuffleNet	2.279M	**151.972M**	9.539
SqueezeNet	**1.235M**	747.099M	**12.514**
MobileNet	2.235M	313.479M	10.155
PydMobileNet	**1.86M**	**195.440M**	8.336

of PydMobileNet is lower than MobileNetV2 because it uses parallel DWconvs with different-size kernels.

As shown in Table 12, the processed Batch-Images per second (BPS) of the ABI-DSM system decreases compared to FPS of Table 11, and the BPS of InceptionNet and ResNet base significantly decline. The BPS of PydMobileNet is also lower than that of MobileNetV2 on the ABI-DSM system. Through the ABI-DSM system, PydMobileNet can process three images batch simultaneously. Then, the processed FPS (PFPS) of the ABI-DSM system based on PydMobileNet is similar to MobileNetV2. The average Precision of the ABI-DSM system based on PydMobileNet is similar to MobileNetV2, as shown in Table 13.

TABLE 12. BPS and PFPS of the ABI-DSM system.

CNN base	BPS	Avg. Batch Size	PFPS
AlexNet	10.203	2.08	**21.234**
InceptionNet	2.1252	8	17.002
ResNet	1.6959	8	13.567
ShuffleNet	9.8890	2.14	21.196
SqueezeNet	**10.091**	2.09	21.177
MobileNet	**10.356**	2.05	**21.243**
PydMobileNet	7.090	2.99	21.186

As shown in Table 12, the BPS of the ABI-DSM system based on InceptionNet and ResNet with the high FLOPs decrease significantly compared to Table 11. The average size of the Batch-Image is more than eight. This results in frame loss because the ABI-DSM system restricts the maximum size of the Batch-Image to eight to prevent an extreme increase in the processing time. The overall processing time of the ABI-DSM system based on the other models increases, but the PFPS of the system is above 20 FPS. This shows that the system can use all of the captured images at about 20 FPS for recognizing the driver's status due to the use of the Batch-Image.

TABLE 13. Precision, Recall, and F1-score of the ABI-DSM system for Test Dataset.

CNN Models	Distraction			Fatigue			Drowsiness			Total Avg.
	P	R	F	P	R	F	P	R	F	P
AlexNet	78.3	82.8	79.2	69.1	62.1	60.9	79.3	74.2	73.4	75.5
InceptionNet	**96.0**	91.9	93.8	**90.6**	90.1	90.2	**94.4**	87.8	90.6	**93.6**
ResNet	95.1	75.9	81.2	83.9	79.6	80.5	91.0	79.2	83.1	89.9
ShuffleNet	86.5	75.3	78.1	68.7	67.5	62.9	72.5	77.0	65.5	75.9
SqueezeNet	71.0	59.8	57.3	54.7	53.1	42.1	60.2	56.9	48.0	62.0
MobileNetV2	91.1	96.1	93.3	**89.7**	92.7	90.6	**93.2**	97.6	95.2	**91.3**
PydMobileNet	**95.4**	95.0	95.1	87.5	89.3	87.6	91.0	94.2	92.4	**91.3**

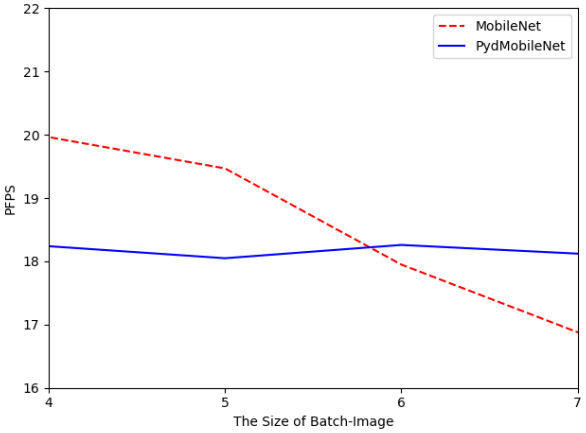


FIGURE 9. The PFPS according to the Size of Batch-Image.

Although the ABI-DSM system based on InceptionNet has the highest average Precision for three tasks, as shown in Table 13, it can not operate in real-time. The Precision of the systems based on AlexNet, ShuffleNet, and SqueezeNet among models capable of real-time operation is lower than the average of 80 percent. Systems based on MobileNet and PydMobileNet are relatively stable in terms of both Precision and FPS. The ABI-DSM system based on PydMobileNet has lower PFPS than that of a MobileNetV2 based system. However, for a Batch-Image size greater than six, the PydMobileNet based system is faster than the MobileNetV2 based system, as shown in Figure 9, which shows PFPS when using a fixed-size Batch-Image.

V. CONCLUSION

This paper proposed the Adaptive Batch-Image based Driver Status Monitoring (ABI-DSM) system for real-time operation on light-weight SBC with GPU. Designs of ABI-DSM focus on reducing parameters of Deep Learning models in the system to process the batch images in real-time, with the following approach. By using facial behavior information, the number of parameters of LSTM, which recognize the driver's status from time-series data, decreases. Also, the facial behavior recognition model is simplified by not detecting and tracking faces in the image. This system uses PydMobileNet, which is an advanced version of MobileNetV2. Parameters of the ABI-DSM based on PydMobileNet is about 73% of the DSM system with the sequence of MobileNetV2 and LSTM.

In the experiment results, the ABI-DSM system based on InceptionNet outperforms others in terms of Precision. However, the size of Batch-Image continuously increases without batch size limitation, then the system eventually terminated. It is a challenge to operate the InceptionNet based ABI-DSM system on light-weight SBC with GPU in real-time. The ABI-DSM systems based on MobileNetV2 and PydMobileNet show reliable FPS and Precision. Among them, MobileNetV2 is more suitable for real-time use on the Jetson Nano.

In cases, which the ABI-DSM system can not occupy 100% GPUs due to the other programs on the same SBC, PydMobileNet is more efficient due to the increase of Batch-Image size.

Although this work considers commercialization, there still are practical issues according to requirements in the industry, such as improving accuracy and reducing the commercializing cost. The vision-based system has limitations in inferring the driver's intentions. The driver's intention is one of the main factors to recognize the status of the driver in the real-world. Many researchers recommend considering traffic context, vehicle dynamics, and driver behaviors together for providing a robust driver assistant system [3], [45]. In the future, the ABI-DSM system will be improved by fusing vehicular, physiological, and vision sensors. Meanwhile, the ABI-DSM system will be implemented and evaluated on cheaper light-weight SBCs, such as the Raspberry Pi 4 and Jetson Nano board with 2GB memory.

REFERENCES

[1] W. Kong, W. Lin, F. Babiloni, S. Hu, and G. Borghini, "Investigating driver fatigue versus alertness using the granger causality network," *Sensors*, vol. 15, no. 8, pp. 19181–19198, Aug. 2015, doi: 10.3390/s150819181.

[2] T. O. Zander, L. M. Andreessen, A. Berg, M. Bleuel, J. Pawlitzki, L. Zawallich, L. R. Krol, and K. Gramann, "Evaluation of a dry EEG system for application of passive brain-computer interfaces in autonomous driving," *Frontiers Human Neurosci.*, vol. 11, pp. 1–16, Feb. 2017, doi: 10.3389/fnhum.2017.00078.

[3] A. Sahayadhas, K. Sundaraj, and M. Murugappan, "Detecting driver drowsiness based on sensors: A review," *Sensors*, vol. 12, no. 12, pp. 16937–16953, Dec. 2012, doi: 10.3390/s121216937.

[4] L. Keyong, J. Lisheng, J. Yuying, X. Huacai, and G. Linlin, "Effects of driver behavior style differences and individual differences on driver sleepiness detection," *Adv. Mech. Eng.*, vol. 7, no. 4, pp. 1–8, 2015, doi: 10.1177/1687814015578354.

[5] M. Ingre, T. Akerstedt, B. Peters, A. Anund, and G. Kecklund, "Subjective sleepiness, simulated driving performance and blink duration: Examining individual differences," *J. Sleep Res.*, vol. 15, no. 1, pp. 47–53, Mar. 2006, doi: 10.1111/j.1365-2869.2006.00504.x.

[6] Z. Nikolić, "Embedded vision in advanced driver assistance systems," in *Advances in Embedded Computer Vision* (Advances in Computer Vision and Pattern Recognition). Cham, Switzerland: Springer, 2014, pp. 45–69, doi: 10.1007/978-3-319-09387_3.

[7] P. Napoletano, "Hand-crafted vs learned descriptors for color texture classification," in *Proc. Int. Workshop Comput. Color Imag.*, 2017, pp. 259–271.

[8] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Proc. 12th Int. Conf. Pattern Recognit.*, vol. 1, Oct. 1994, pp. 582–585.

[9] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognit.*, vol. 29, no. 1, pp. 51–59, 1996.
[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1150–1157, doi: 10.1109/ICCV.1999.790410.
[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, pp. 511–518, doi: 10.1109/CVPR.2001.990517.
[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.
[13] L. Breiman, J. Friedman, H. J. R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Philadelphia, PA, USA: Wadsworth Brooks/Cole Advanced Books Software.
[14] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992, doi: 10.1080/00031305.1992.10475879.
[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[16] J. Walsh, N. O. Mahony, S. Campbell, A. Carvalho, L. Krpalkova, G. Velasco-Hernandez, S. Harapanahalli, and D. Riordan, "Deep learning vs traditional computer vision," in *Advances in Computer Vision (CVC)* (Advances in Intelligent Systems and Computing), vol. 943. Cham, Switzerland: Springer, 2019, pp. 128–144, doi: 10.1007/978-3-030-17795-9_10.
[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, no. 2, pp. 1097–1105, doi: 10.1145/3065386.
[18] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5379–5390, Jun. 2019, doi: 10.1109/TVT.2019.2908425.
[19] W. Kim, W.-S. Jung, and H. K. Choi, "Lightweight driver monitoring system based on multi-task mobilenets," *Sensors*, vol. 19, no. 14, p. 3200, Jul. 2019, doi: 10.3390/s19143200.
[20] M. Bologna, R. Agostino, B. Gregori, D. Belvisi, D. Ottaviani, C. Colosimo, G. Fabbrini, and A. Berardelli, "Voluntary, spontaneous and reflex blinking in patients with clinically probable progressive supranuclear palsy," *Brain*, vol. 132, no. 2, pp. 502–510, Jun. 2008, doi: 10.1093/brain/awn317.
[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[22] V.-T. Hoang and K.-H. Jo, "PydMobileNet: Pyramid depthwise separable convolution networks for image classification," in *Proc. IEEE 28th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2019, pp. 1430–1434, doi: 10.1109/ISIE.2019.8781130.
[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861
[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, Jun. 2018, pp. 4510–4520.[Online]. Available: https://arxiv.org/abs/1801.04381
[25] S. Park, F. Pan, S. Kang, and C. D. Yoo, "Driver drowsiness detection system based on feature representation learning using various deep networks," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, vol. 10118. Cham, Switzerland: Springer, 2016, pp. 154–164, doi: 10.1007/978-3-319-54526-4_12.
[26] B. Reddy, Y.-H. Kim, S. Yun, C. Seo, and J. Jang, "Real-time driver drowsiness detection for embedded system using model compression of deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 438–445, doi: 10.1109/CVPRW.2017.59.
[27] R. Naqvi, M. Arsalan, G. Batchuluun, H. Yoon, and K. Park, "Deep learning-based gaze detection system for automobile drivers using a NIR camera sensor," *Sensors*, vol. 18, no. 2, p. 456, Feb. 2018, doi: 10.3390/s18020456.
[28] B. Ahn, D.-G. Choi, J. Park, and I. S. Kweon, "Real-time head pose estimation using multi-task deep neural network," *Robot. Auto. Syst.*, vol. 103, pp. 1–12, May 2018, doi: 10.1016/j.robot.2018.01.005.
[29] J. W. Baek, B.-G. Han, K.-J. Kim, Y.-S. Chung, and S.-I. Lee, "Real-time drowsiness detection algorithm for driver state monitoring systems," in *Proc. 10th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2018, pp. 73–75, doi: 10.1109/ICUFN.2018.8436988.
[30] Q. Massoz, J. G. Verly, and M. Van Droogenbroeck, "Multi-timescale drowsiness characterization based on a video of a driver's face," *Sensors*, vol. 18, no. 9, p. 2801, 2018, doi: 10.3390/s18092801.
[31] D. Tran, H. Manh Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," *IET Intell. Transp. Syst.*, vol. 12, no. 10, pp. 1210–1219, Dec. 2018, doi: 10.1049/iet-its.2018.5172.
[32] L. Celona, L. Mammana, S. Bianco, and R. Schettini, "A multi-task CNN framework for driver face monitoring," in *Proc. IEEE 8th Int. Conf. Consum. Electron.-Berlin (ICCE-Berlin)*, Sep. 2018, pp. 1–4, doi: 10.1109/ICCE-Berlin.2018.8576244.
[33] W. Liu, J. Qian, Z. Yao, X. Jiao, and J. Pan, "Convolutional two-stream network using multi-facial feature fusion for driver fatigue detection," *Future Internet*, vol. 11, no. 5, p. 115, 2019, doi: 10.3390/fi11050115.
[34] Y. Ji, S. Wang, Y. Zhao, J. Wei, and Y. Lu, "Fatigue state detection based on multi-index fusion and state recognition network," *IEEE Access*, vol. 7, pp. 64136–64147, 2019, doi: 10.1109/ACCESS.2019.2917382.
[35] W. Deng and R. Wu, "Real-time driver-drowsiness detection system using facial features," *IEEE Access*, vol. 7, pp. 118727–118738, 2019, doi: 10.1109/ACCESS.2019.2936663.
[36] Q. Cheng, W. Wang, X. Jiang, S. Hou, and Y. Qin, "Assessment of driver mental fatigue using facial landmarks," *IEEE Access*, vol. 7, pp. 150423–150434, 2019, doi: 10.1109/ACCESS.2019.2947692.
[37] F. You, X. Li, Y. Gong, H. Wang, and H. Li, "A real-time driving drowsiness detection algorithm with individual differences consideration," *IEEE Access*, vol. 7, pp. 179396–179408, 2019, doi: 10.1109/ACCESS.2019.2958667.
[38] Y.-C. Tsai, P.-W. Lai, P.-W. Huang, T.-M. Lin, and B.-F. Wu, "Vision-based instant measurement system for driver fatigue monitoring," *IEEE Access*, vol. 8, pp. 67342–67353, 2020, doi: 10.1109/ACCESS.2020.2986234.
[39] K. Li, Y. Gong, and Z. Ren, "A fatigue driving detection algorithm based on facial multi-feature fusion," *IEEE Access*, vol. 8, pp. 101244–101259, 2020, doi: 10.1109/ACCESS.2020.2998363.
[40] K. S. A. A. Zisserman, "Very deep convolutional networks for largescale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556v6
[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9.
[42] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "SqueezeNext: Hardware-aware neural network design," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPR)*, 2018, pp. 1638–1647.
[43] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[44] L. Sifre and M. Stéphane, "Rigid-motion scattering for image classification," 2014, *arXiv:1403.1687*. [Online]. Available: https://arxiv.org/abs/1403.1687
[45] Y. Xing, C. Lv, H. Wang, H. Wang, Y. Ai, D. Cao, E. Velenis, and F.-Y. Wang, "Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4377–4390, May 2019, doi: 10.1109/TVT.2019.2903299.

WHUI KIM received the B.S. degree in electronics and information engineering from Korea University, in 2014, and the M.S. degree in computer software from the University of Science and Technology (UST), in 2016. He joined ETRI, in 2016. He is currently a Researcher with the Intelligent Robotics Research Division, ETRI. His research interests include deep learning and machine learning.

YOUL-KYEONG LEE (Member, IEEE) received the B.S. degree in electronics engineering from the University of Ulsan, Ulsan, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the Graduate School of Electrical Engineering. His research interests include image processing, pattern recognition, computer vision, and machine learning. He is a member of societies, such as the IEEE and the IEEE Industrial Electronics Society.

WOO-SUNG JUNG (Member, IEEE) received the B.S. (dual degree) in electrical and computer engineering, information and computer engineering, and the M.S. and Ph.D. degrees from the School of Computer Engineering, Ajou University, South Korea, in 2007, 2009, and 2015, respectively. He is currently a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His current research interests include wireless networking, the Internet of Things, device-to-device communication, and embedded systems.

DAESEUNG YOO received the B.S., M.S., and Ph.D. degrees from the Department of Computer Engineering and Information Technology, University of Ulsan, in 1998, 2001, and 2011, respectively. Since 2009, he has been a Research Engineer with the Intelligent Robotics Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His current research interests include applied software engineering, mobile communication, ship-IT convergence, and ad-hoc networks.

DO-HYUN KIM received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Pusan National University, South Korea, in 1995, 1997, and 2016, respectively. Since 2000, he has been a Research Member with the Electronics and Telecommunications Research Institute (ETRI). His research interests include automotive, database, ubiquitous sensor networks, and ITS.

KANG-HYUN JO (Senior Member, IEEE) received the B.S. degree from Busan National University, South Korea, in 1989, and the M.S. and Ph.D. degrees in computer controlled machinery from Osaka University, Japan, in 1993 and 1997, respectively. He joined the School of Electrical Engineering, University of Ulsan, Ulsan, South Korea. He is currently serving as the Faculty Dean of the School of Electrical Engineering, University of Ulsan. His research interests include computer vision, robotics, autonomous vehicle, and ambient intelligence. He has also been involved in organizing many international conferences, such as the International Workshop on Frontiers of Computer Vision, the International Conference on Intelligent Computation, the International Conference on Industrial Technology, the International Conference on Human System Interactions, and the Annual Conference of the IEEE Industrial Electronics Society. He is an Editorial Board Member of the International Journal of Control, Automation, and Systems and a Vice President for Membership of ICROS. He has served as a Director or an AdCom Member of the Institute of Control, Robotics and Systems, The Society of Instrument and Control Engineers, and the IEEE IES Technical Committee on Human Factors Chair, AdCom member, and the Secretary, in 2019.