Driver Behavior Recognition via Interwoven Deep Convolutional Neural Nets with Multi-stream Inputs

Chaoyun Zhang[♣], Rui Li[♣], Woojin Kim[♦], Daesub Yoon[♦], and Paul Patras[♣]

Abstract*—Recognizing driver behaviors is becoming vital for in-vehicle systems that seek to reduce the incidence of car accidents rooted in cognitive distraction. In this paper, we harness the exceptional feature extraction abilities of deep learning and propose a dedicated Interwoven Deep Convolutional Neural Network (InterCNN) architecture to tackle the accurate classification of driver behaviors in real-time. The proposed solution exploits information from multi-stream inputs, i.e., in-vehicle cameras with different fields of view and optical flows computed based on recorded images, and merges through multiple fusion layers abstract features that it extracts. This builds a tight ensembling system, which significantly improves the robustness of the model. We further introduce a temporal voting scheme based on historical inference instances, in order to enhance accuracy. Experiments conducted with a real world dataset that we collect in a mock-up car environment demonstrate that the proposed InterCNN with MobileNet convolutional blocks can classify 9 different behaviors with 73.97% accuracy, and 5 aggregated behaviors with 81.66% accuracy. Our architecture is highly computationally efficient, as it performs inferences within 15 ms, which satisfies the real-time constraints of intelligent cars. In addition, our InterCNN is robust to lossy input, as the classification remains accurate when two input streams are occluded.

***Index Terms*—Driver behavior recognition, deep learning, convolutional neural networks**

I. Introduction

DRIVER's cognitive distraction is a major cause of unsafe driving, which leads to severe car accidents every year [1]. Actions that underlie careless driving include interacting with passengers, using a mobile phone (e.g., for text messaging, game playing, and web browsing), and consuming food or drinks [2]. Such behaviors contribute significantly to delays in driver's response to unexpected events, thereby increasing the risk of collisions. Identifying drivers' behaviors is therefore becoming increasingly important for car manufacturers, who aim to build in-car intelligence that can improve safety by notifying drivers in real-time of potential hazards. Further, although full car automation is years ahead, inferring driver's behaviour is essential for vehicles with partial ("hands off") and conditional ("eyes off") automation, which will dominate the market at least until 2030 [3]. This is because the driver must either be ready to take control at any time or intervene in situations where the vehicle cannot complete certain critical functions [4].

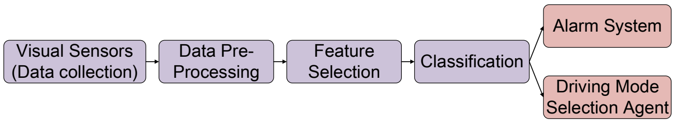


Fig. 1: The typical pipeline of driver behavior classification and alarm/driving mode selection systems.

Modern driver behavior classification systems usually rely on videos acquired from in-vehicle cameras, which record the movements and the facial expressions of the driver [5]. The videos captured are routinely partitioned into sequences of image frames, which are then pre-processed for features selection [6]. Such features are fed to pre-trained classifiers to perform identification of different actions that the driver performs. Subsequently, the classifier may trigger an alarm system in manual driving cars or provide input to a driving mode selection agent in semi-autonomous vehicles. We illustrate this pipeline in Fig. 1. During this process, the accuracy of the classifier is directly related to the performance of the system. In addition, the system should perform such classification in real-time, so as to help the driver mitigate unsafe circumstances in a timely manner. Achieving high accuracy while maintaining runtime efficiency is however challenging, yet striking appropriate trade-offs between these aims is vital for intelligent and autonomous vehicles.

Underpinned by recent advances in parallel computing, deep neural networks [7] have achieved remarkable results in various areas, including computer vision [8], control [9], and autonomous driving [10], [11], as they can automatically extract features from raw data without requiring expensive hand-crafted feature engineering. Graphics processing units (GPUs) allow to train deep neural networks rapidly and with great accuracy, and perform inferences fast. Moreover, System on Chip (SoC) designs optimized for mobile artificial intelligence (AI) applications are becoming more powerful and computationally efficient [12], and embedding deep learning in car systems increasingly affordable [13]. Therefore, potential exists to build high precision driver behavior classification systems without compromising runtime performance.

In this paper, we design a driver behavior recognition system that uniquely combines different Convolutional Neural Network (CNN) structures, to accurately perform this task in real-time. As such, we make the following **key contributions**:

[♣] C. Zhang, R. Li, and P. Patras are with the Institute for Computing Systems Architecture (ICSA), School of Informatics, University of Edinburgh, UK. Emails: {chaoyun.zhang, rui.li, paul.patras}@ed.ac.uk.

[♦] W. Kim and D. Yoon are with the Electronics and Telecommunications Research Institute (ETRI), Daejon, South Korea. Emails: {wjinkim, eyetracker}@etri.re.kr.

1) We build a mock-up environment to emulate self-driving car conditions and instrument a detailed user study for data collection purposes. Specifically, we deploy side and front facing cameras to record the body movements and facial expressions of 50 participant drivers, throughout a range of tasks they performed. This leads to a large driver behavior video dataset, which we use for model training and evaluation.
2) We architect original Interwoven Convolutional Neural Networks (InterCNNs) to perform feature extraction and fusions across multiple levels, based on multi-stream video inputs and optical flow information. Our design allows to plug in different lightweight CNN architectures (e.g. MobileNet [14], [15]) to improve the computation efficiency of in-vehicle systems.
3) We demonstrate that our InterCNNs with MobileNet blocks and a temporal voting scheme, which enhances accuracy by leveraging historical inferences, can classify 9 different behaviors with 73.97% accuracy, and 5 aggregated behaviors (i.e., grouping tasks that involve the use of a mobile device, and eating & drinking) with 81.66% accuracy. Our architecture can make inferences within 15 ms, which satisfies the timing constraints posed by real car systems. Importantly, our architecture is highly robust to lossy input, as it remains accurate when two streams of the input are occluded.

The results obtained demonstrate the feasibility of accurate inference of driver's behavior in real-time, making important steps towards fulfilling the multi-trillion economic potential of the driverless car industry [16].

The rest of the paper is organized as follows. In Sec. II we discussed relevant related work. In Sec. III we present our data collection and pre-processing efforts, which underpin the design of our neural network solution that we detail in Sec. IV-A. We demonstrate the performance of the proposed InterCNNs by means of experiments reported in Sec. V. Sec. VI concludes our contributions.

II. Related Work

The majority of the driver behavior classification systems are based on in-vehicle vision instruments (i.e., cameras or eye-tracking devices), which constantly monitor the movements of the driver [17]. The core of such systems is therefore tasked with a computer vision problem, whose objective is to classify actions performed by drivers, using sequences of images acquired in real-time. Existing research can be categorized into two main classes: non deep learning approaches and deep learning approaches.

A. Non Deep Learning Based Driver Behavior Identification

In [18], Liu *et al.* employ Laplacian Support Vector Machine (SVM) and extreme learning machine techniques to detect drivers' distraction, using labelled data that captures vehicle dynamic and drivers' eye and head movements. Experiments show that this semi-supervised approach can achieve up to 97.2% detection accuracy. Li *et al.* pursue distraction detection from a different angle. They exploit kinematic signals from the vehicle Controller Area Network (CAN) bus, to reduce the dependency on expensive vision sensors. Detection is then performed with an SVM, achieving 95% accuracy.

Ragab *et al.* compare the prediction accuracy of different machine learning methods in driving distraction detection [19], showing that Random Forests perform best and require only 0.05 s per inference. Liao *et al.* consider drivers' distraction in two different scenarios, i.e., stop-controlled intersections and speed-limited highways [1]. They design an SVM classifier operating with Recursive Feature Elimination (RFE) to detect driving distraction. The evaluation results suggest that by fusing eye movements and driving performance information, classification accuracy can be improved in stop-controlled intersection settings.

B. Deep Learning Based Driver behavior Identification

Deep learning is becoming increasingly popular for identifying driver behaviors. In [20], a multiple scale Faster Region CNN is employed to detect whether a driver is using a mobile phone or their hands are on the steering wheel. The solution operates on images of the face, hands and steering wheel separately, and then performs classification on these regions of interest. Experimental results show that this model can discriminate behaviors with high accuracy in real-time. Majdi *et al.* design a dedicated CNN architecture called Drive-Net to identify 10 different behaviors of distracted driving [21]. Experiments suggest that applying Region of Interest (RoI) analysis on images of faces can significantly improve accuracy.

Tran *et al.* build a driving simulator named Carnetsoft to collect driving data, and utilize 4 different CNN architectures to identify 10 distracted and non-distracted driving behaviors [22]. The authors observe that deeper architectures can improve the detection accuracy, but at the cost of increased inference times. Investigating the trade-off between accuracy and efficiency remains an open issue. Yuen *et al.* employ a CNN to perform head pose estimation during driving [23]. Evaluation results suggest that incorporating a Stacked Hourglass Network to estimate landmarks and refine the face detection can significantly improve the accuracy with different occlusion levels. In [24], Streiffer *et al.* investigate mixing different models, i.e., CNNs, recurrent neural networks (RNNs), and SVMs, to detect driver distraction. Their ensembling CNN + RNN approach significantly outperforms simple CNNs in terms of prediction accuracy.

Recognizing driver's behavior with high accuracy, using inexpensive sensing infrastructure, and achieving this in real-time remains challenging, yet mandatory for intelligent vehicles that can improve safety and reduce the time during which the driver is fully engaged. To the best of our knowledge, existing work fails to meet all these requirements.

III. Data Collection and Pre-Processing

In this work, we propose an original driver behavior recognition system that can classify user actions accurately in real-time, using input from in-vehicle cameras. Before delving into our solution (Sec. IV-A), we discuss the data collection and pre-processing campaign that we conduct while mimicking

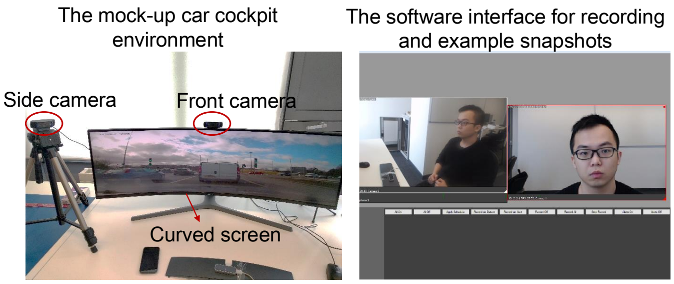

The mock-up car cockpit environment
The software interface for recording and example snapshots
Side camera
Front camera
Curved screen

Fig. 2: Mock-up car cockpit environment (left) used for data collection and snapshots of the software interface employed for video capture (right). The curved screen shows a driving video to the participant, while the side and front cameras record their body movements and facial expressions, respectively.

an autonomous vehicle environment, in order to facilitate the design, training, and evaluation of our neural network model.

A. Data Collection

We set up the mock-up car cockpit environment illustrated in Fig. 2 and conduct a user behavior study, whereby we emulate automated driving conditions and record data from two cameras (one to the side of the driver, the other front-facing) that capture driver's actions. We recruit a total of 50 participants, 72% male and 38% female, with different age, years of driving experience, first spoken language, and level of computer literacy. During the experiments, we use a 49-inch Ultra High Definition monitor, onto which each participant is shown 35-minute of 4K dashcam footage of both motorway and urban driving, while being asked to alternate between 'driving' and performing a range of tasks. The cameras record the behavior of the participants from different angles, capturing body movements and facial expressions with 640×360 per-frame pixel resolution and frame rate ranging between 17.14 and 24.74 frames per second (FPS).

We use the OpenCV vision library [25] together with Python, in order to label each of the frames of the approximately 60 hours of video recorded, distinguishing between the following actions:

1) **Normal driving:** The participant focuses on the road conditions shown on the screen and acts as if driving.
2) **Texting:** The participant uses a mobile phone to text messages to a friend.
3) **Eating:** The participant eats a snack.
4) **Talking:** The participant is engaged in a conversation with a passenger.
5) **Searching:** The participant is using a mobile phone to find information on the web through a search engine.
6) **Drinking:** The participant serves a soft drink.
7) **Watching video:** The participant uses a mobile phone to watch a video.
8) **Gaming:** The participant uses a mobile phone to play a video game.
9) **Preparing:** The participant gets ready to begin driving or finishes driving.

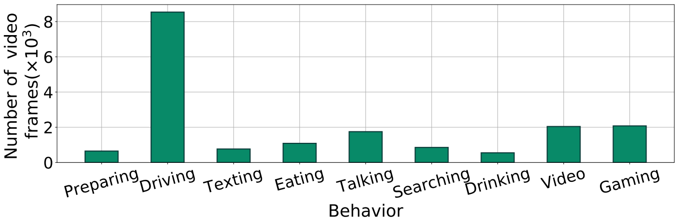


Fig. 3: Summary of the total amount of data (video frames) collected for each driver behavior.

In each experiment, the participant was asked to perform actions (2)–(8) once, while we acknowledge that in real-life driving such behaviors can occur repeatedly. Fig. 3 summarizes the amount of data (number of video frames) collected for each type of action that the driver performed.

B. Data Pre-processing

Recall that the raw videos recorded have 640×360 resolution. Using high-resolution images inevitably introduces storage, computational, and data transmission overheads, which would complicate the model design. Therefore, we employ fixed bounding boxes to crop all videos, in order to remove the background, and subsequently re-size the videos to obtain lower resolution versions. Note that the shape and position of the bounding boxed adopted differ between the videos recorded with side and front cameras. We illustrate this process in Fig. 4.

Adding Optical Flow (OF) [26] to the input of a model has proven effective in improving accuracy [27]. The OF is the instantaneous velocity of the moving objects under scene surface. It can reflect the relationship between the previous and current frames, by computing the changes of the pixel values between adjacent frames in a sequence. Therefore, OF can explicitly describe the short-term motion of the driver, without requiring the model to learn about it. The OF vector $\mathbf{d}^{(x,y)}$ at point (x, y) can be decomposed into vertical and horizontal components, i.e., $\mathbf{d}^{(x,y)} = \{d_v^{(x,y)}, d_h^{(x,y)}\}$. It has the same resolution as the original images, as the computation is

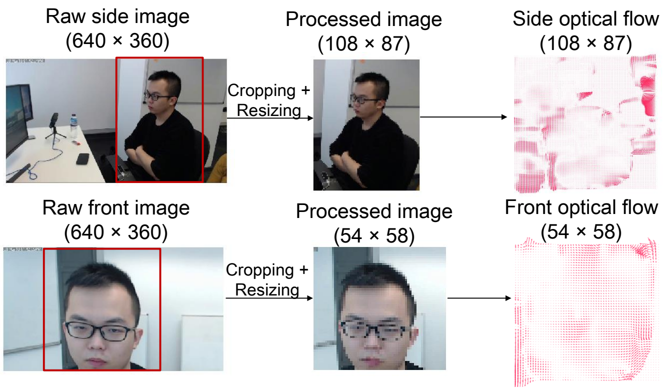
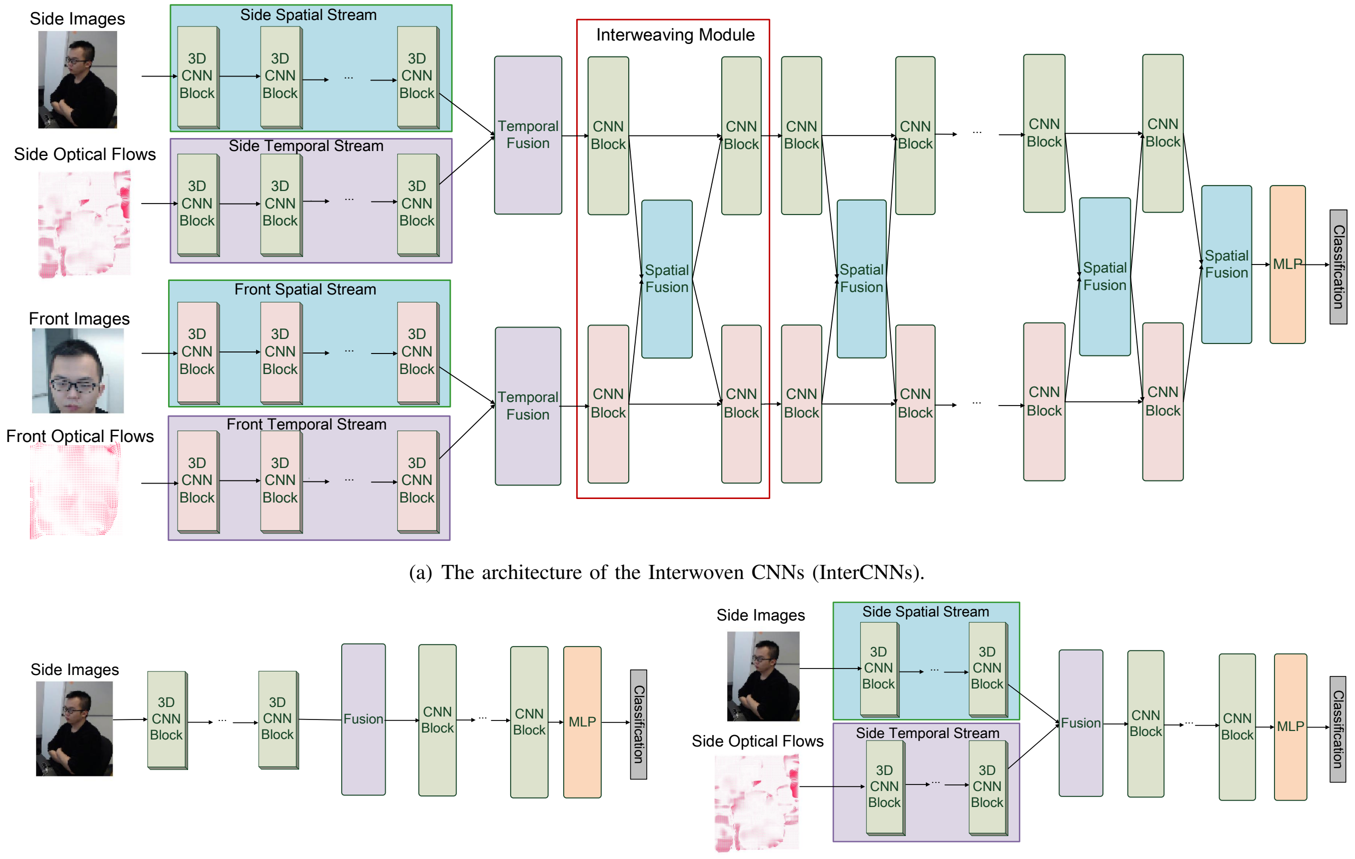

Raw side image (640 × 360)
Cropping + Resizing
Processed image (108 × 87)
Side optical flow (108 × 87)
Raw front image (640 × 360)
Cropping + Resizing
Processed image (54 × 58)
Front optical flow (54 × 58)

Fig. 4: The data pre-processing procedure (cropping and resizing) and optical flow quiver plots.



(a) The architecture of the Interwoven CNNs (InterCNNs).

(b) The architecture of a plain CNN.

(c) The architecture of a two-stream CNN (TS-CNN).

Fig. 5: Three different neural network architectures employed in this study. The plain CNN only uses the side video stream as input, the two-streaming CNN adds extra side OFs, while the InterCNN employs both side and front video streams and optical flows.

done pixel-by-pixel. We show an example of the OF in Fig. 4. Our classifier will use OF information jointly with labelled video frames as the input. The experiments we report in Sec. V confirm that indeed this improves the inference accuracy.

Lastly, we downsample the videos collected, storing only every third frame and obtaining a dataset with 5.71–8.25 FPS, which reduces data redundancy. We will feed the designed model with 15 consecutive video frames and corresponding 14 OF vectors, spanning 1.82 to 2.62 seconds of recording. Such duration has been proven sufficient to capture entire actions, while obtaining satisfactory accuracy [28].

IV. Multi-stream Interwoven CNNs

We design a deep neural network architecture, named Interwoven CNN (InterCNN) that uses multi-stream inputs (i.e., side video streams, side optical flows, front video streams, and front optical flows) to perform driver behavior recognition. We illustrate the overall architecture of our model in Fig. 5(a). For completeness, we also show two simpler architectures, namely *(i)* a plain CNN, which uses only the side video stream as input (see Fig. 5(b)); and a *(ii)* two-stream CNN (TS-CNN), which takes the side video stream and the side optical flow as input (see Fig. 5(c)). Both of these structures can be viewed as components of the InterCNN.

A. The InterCNN Architecture

Diving into Fig. 5(a), the InterCNN is a hierarchical architecture which embraces multiple types of blocks and modules. It takes four different streams as input, namely side video stream, side optical flow, front video stream and front optical flow. Note that these streams are all four-dimensional, i.e., (time, height, width, RGB channels) for each video frame, and (time, height, width, vertical and horizontal components) for OF frames. The raw data is individually processed in parallel by 7 stacks of 3D CNN blocks. A 3D CNN block is comprised of a 3D convolutional layer to extract spatio-temporal features [29], a Batch Normaliazation (BN) layer for training acceleration [30], and a Scaled Exponential Linear Unit (SELU) activation function to improve the model non-linearity and representability [31]. Here,

$$\mathbf{SELU}(\mathbf{x}) = \lambda \begin{cases} \mathbf{x}, & \text{if } \mathbf{x} > 0; \\ \alpha e^{\mathbf{x}} - \alpha, & \text{if } \mathbf{x} \leq 0, \end{cases}$$

where the parameters $\lambda = 1.0507$ and $\alpha = 1.6733$ are frequently used. We refer to these four streams of 3D CNNs as side spatial stream, side temporal stream, front spatial stream, and front temporal stream respectively, according to the type of input handled. Their outputs are passed to two temporal

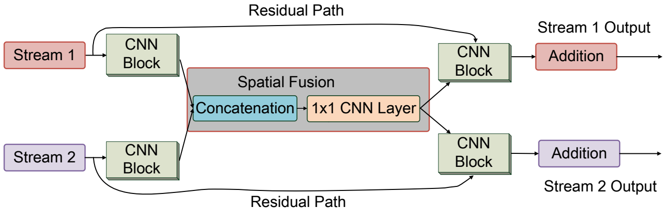


Fig. 6: The anatomic structure of an Interweaving module.

fusion layers to absorb the time dimension and perform the concatenation operation along the channel axis. Through these temporal fusion layers, intermediate outputs of spatial and temporal streams are merged and subsequently delivered to 25 stacks of Interweaving modules. We illustrate the construction of such modules in Fig. 6 and detial their operation next.

B. Interweaving Modules

The Interweaving module draws inspiration from ResNets [32] and can be viewed as a multi-stream version of deep residual learning. The two inputs of the module are processed by different CNN blocks individually, and subsequently delivered to a spatial fusion layer for feature aggregation. The spatial fusion layer comprises a concatenation operation and a 1×1 convolutional layer, which can reinforce and tighten the overall architecture, and improve the robustness of the model [33]. Experiments will demonstrate that this enables the model to maintain high accuracy even if the front camera is blocked completely. After the fusion, another two CNN blocks will decompose the merged features in parallel into two-stream outputs again. This maintains the information flow intact. Finally, the residual paths connect the inputs and the outputs of the final CNN blocks, which facilitates fast backpropagation of the gradients during model training. These paths also build ensembling structures with different depths, which have been proven effective in improving inference accuracy [34]. After processing by the Interweaving blocks, the intermediate outputs obtained are sent to a Multi-Layer Perceptron (MLP) to perform the final classification.

C. CNN Blocks Employed

The CNN blocks employed within the interweaving modules are key to performance, both in terms of accuracy and inference time. Our architecture is sufficiently flexible to allow different choices for these CNN blocks. In this work, we explore the vanilla CNN block, MobileNet [14], and MobileNet V2 [15] structures, and compare their performance. We show the architectures of these choices in Fig. 7.

The vanilla CNN block embraces a standard 2D CNN layer, a BN layer and a Rectified Linear Unit (ReLU) activation function. This is a popular configuration and has been employed in many successful classification architectures, such as ResNet [32]. However, the operations performed in a CNN layer are complex and involve a large number of parameters. This may not satisfy the resource constraints imposed by vehicular systems. The MobileNet [14] decomposes the traditional CNN layer into a depthwise convolution and a pointwise convolution, which significantly reduces the number of parameters required. Specifically, depthwise convolution employs single a convolutional filter to perform computations on individual input channels. Thereby, this generates an intermediate output that has the same number of channels as the input. The outputs are subsequently passed to a pointwise convolution module, which applies a 1×1 filter to perform channel combination. MobileNet further employs a hyperparameter α to control the number of channels, and ρ to control the shape of the feature maps. We set both α and ρ to 1 in our design. In summary, the MobileNet block breaks the filtering and combining operations of a traditional CNN block into two layers. This significantly reduces the computational complexity and number of parameters required, while improving efficiency in resource-constrained devices.

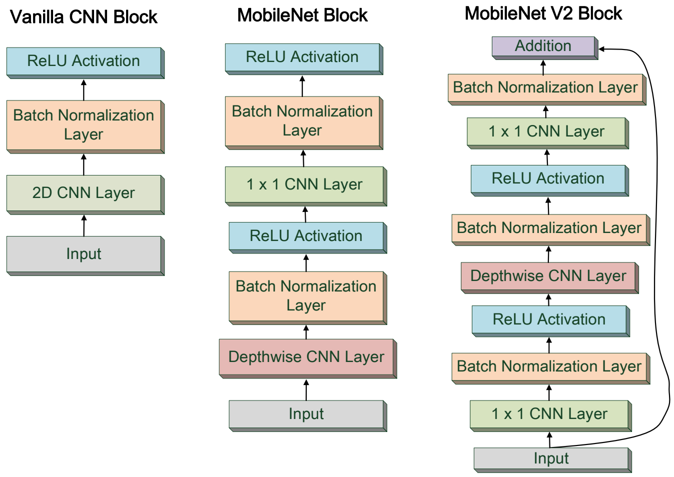


Fig. 7: Three different CNN blocks employed in this study.

The MobileNet V2 structure [15] improves the MobileNet by introducing an inverted residual and linear bottleneck. The inverted residual incorporates the residual learning specific to ResNets. The input channels are expanded through the first 1×1 convolutional layer, and compressed through the depthwise layer. The expansion is controlled by a parameter t, which we set to 6 as default. To reduce information loss, the ReLU activation function in the last layer is removed. Compared to MobileNet, the second version has fewer parameters and higher memory efficiency. As we will see, this architecture may sometimes exhibit superior accuracy. Both MobileNet and MobileNet V2 are tailored to embedded devices, as they make inferences faster with smaller models. These makes them suitable for in-vehicle classification systems.

V. Experiments

In this section, we first describe briefly the implementation of the proposed InterCNN for driver behavior recognition, then compare the prediction accuracy of different CNN blocks that we can incorporate in this architecture. Subsequently, we examine complexity–accuracy tradeoffs, introduce a temporal voting scheme to improve performance, and show that our

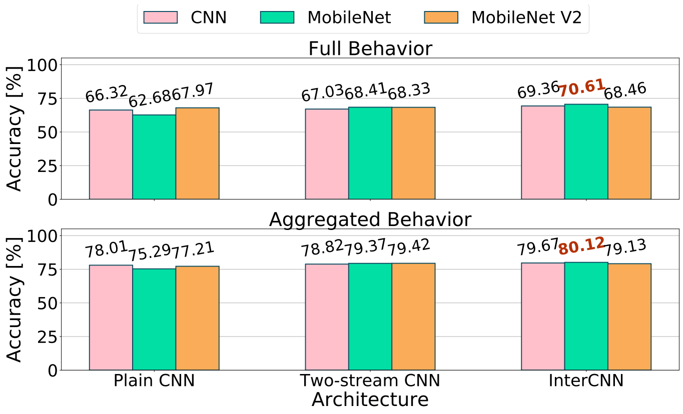


Fig. 8: Prediction accuracy in case of classification among 9 different driver behaviors (top) and aggregate tasks (bottom), for all the neural network architectures considered.

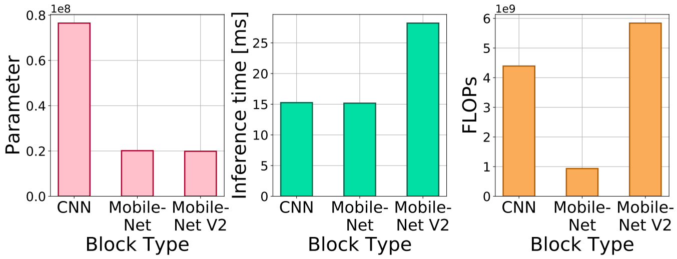


Fig. 9: Comparison of number of parameter (left), inference time (middle) and floating point operations (FLOPs) on InterCNNs with different CNN blocks.

architecture is robust to losses in the input. Finally, we dive deeper into the operation of the proposed model, visualizing the output of the hidden layers, which helps understanding what knowledge is learned by the InterCNN.

A. Implementation

We implement the different neural network architectures studied in TensorFlow [35] with the TensorLayer library [36]. We train all models on a computing cluster equipped with 1-2 NVIDIA TITAN X and Tesla K40M GPUs for approximately 10 days and perform early-stop based on the validation error. For training, we use the Adam optimization algorithm [37], which is based on stochastic gradient descent. With this we seek to minimize the standard cross-entropy loss function between true labels and the outputs produced by the deep neural networks. To maintain consistency, we test all models using an NVIDIA TITAN X GPU when evaluating their computation efficiency.

B. Accuracy Assessment

We randomly partition the entire dataset into a training set (30 videos), a validation set (10 videos) and a test set (10 videos). We assessed the accuracy of our solution on two categories of behaviors. The first considers all the 9 different actions performed by the driver (see Sec. III). In the second, we aggregate the behaviors that are visually similar and carry similar cognitive status. In particular, [Texting, Searching, Watching Video, Gaming] are aggregated into a "Using phone" behavior, and [Eating, Drinking] are combined into a single "Eat & Drink" action.

In Fig. 8, we show the prediction accuracy of the InterCNN architecture with all the CNN block types considered, as well as that of plain and two-stream CNN architectures, each employing the same three types of blocks. Observe that in the case of "full behavior" recognition (top subfigure), the proposed InterCNN with MobileNet blocks achieves the highest prediction accuracy, outperforming the plain CNN by 7.93%. Further, we can see that feeding the neural network with richer information (optical flows and front images) improves accuracy, as our two-stream CNN and InterCNN on average outperform the plain CNN by 2.26% and 3.81% respectively. This confirms that the OF and facial expressions provide useful descriptions of the behaviors, which our architectures effectively exploits. It is also worth noting that, although the performance gains over the plain CNN may appear relatively small, the amount of computational resource required by our architecture, inference times, and complexity are significantly smaller. We will detail these aspects in the following subsection.

Turning attention to the aggregated behavior (bottom subfigure), observe that the accuracy improves significantly compared to when considering all the different actions the driver might perform, as we expect. This is because some behaviors demonstrate similar patterns (e.g., texting and web searching), which makes discriminating among these extremely challenging. Overall, the InterCNN with MobileNet blocks obtains the best prediction performance when similar behaviors are aggregated, outperforming other architectures by up to 4.83%. In addition, our two-stream CNN and InterCNN consistently outperform the plain CNN.

C. Model Complexity & Inference Time

Next, we compare the model size (in terms of number or weights and biases to be configured in the model), inference time, and complexity (quantified through floating point operations – FLOPs) of InterCNNs with different CNN blocks. Lower model size will pose small storage and memory requirements on the in-vehicle system. The inference time refers to how long it takes to perform one instance of driver behavior recognition. This is essential, as such application are required to perform in real-time. Lastly, the number of FLOPs [38] is computed by counting the number of mathematical operation or assignments that involve floating-point numbers, and is routinely used to evaluate the complexity of a model.

We illustrate this comparison in Fig. 9. Observe that MobileNet and MobileNet V2 have similar numbers of parameters, and these are 4 times fewer than those of vanilla CNN blocks. This is consistent with the conclusions drawn in [14] and [15]. In addition, InterCNNs with MobileNet blocks can infer driver behavior within 15 ms per instance (center subfigure) with the help of a GPU, which satisfies the real-time constraints of intelligent vehicle systems. Runtime

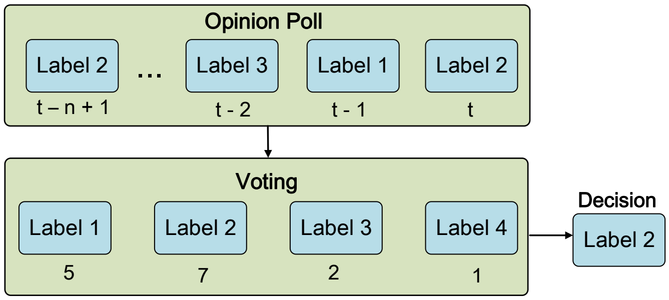


Fig. 10: Illustration of the Temporal Voting (TV) scheme.

performance is indeed similar to that of an architecture employing CNN blocks, yet less complex, while an architecture with MobileNet blocks is 46.2% faster than with MobileNet V2 blocks. Lastly, the number of FLOPs required by an InterCNN with MobileNet blocks is approximately 4.5 and 6 times smaller than when employing CNN and respectively MobileNet V2 blocks. This requirement can be easily handled in real-time even by a modern CPU.

D. Temporal Voting

In the collected dataset, since the videos are recorded at high FPS rate, we observe that the behavior of a driver will not change very frequently over consecutive frames that span less than 3 seconds. Therefore, we may be able to reduce the likelihood of misclassification by considering the actions predicted over recent frames. To this end, we employ a temporal voting scheme, which constructs an opinion poll storing the inferred behaviors over the n most recent frames, and executes a "voting" procedure. Specifically, the driver's action is determined by the most frequent label in the poll. We illustrate the principle of this Temporal Voting (TV) procedure in Fig. 10. We set $n = 15$, by which the poll size bears the same temporal length as the inputs.

We show the prediction accuracy before and after applying TV in Tab I and II. Observe that the TV scheme improves the classification accuracy of all architectures on both full and aggregated behavior sets. In particular, the accuracy on full behavior recognition increases by 1.99%, and that of aggregated behavior recognition by 1.80%. This demonstrates the effectiveness of the proposed TV scheme.

TABLE I: Inference accuracy with different CNN blocks over full behaviors, before\after applying the TV scheme.

Block	Plain CNN	TS-CNN	InterCNN
CNN	66.32%\69.74%	67.03%\68.27%	69.39%\68.40%
MobileNet	62.68%\63.59%	68.41%\70.76%	70.61%**73.97%**
MobileNet V2	67.97%\69.35%	68.33%\70.89%	68.46%\70.61%

TABLE II: Inference accuracy with different CNN blocks over aggregated behaviors before\after applying the TV scheme.

Block	Plain CNN	TS-CNN	InterCNN
CNN block	78.01%\79.70%	78.82%\80.77%	79.67%\81.25%
MobileNet	75.29%\77.78%	79.37%\81.40%	80.62%**81.66%**
MobileNet V2	77.21%\79.14%	79.42%\80.97%	79.13%\80.55%

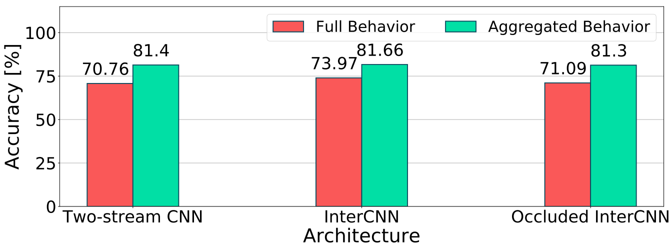


Fig. 11: Comparison of accuracy of two-stream CNN, InterCNN, and InterCNN with occluded inputs. All architectures employ MobileNet blocks.

E. Operation with Lossy Input

In this subsection, we investigate the robustness of our InterCNN to losses in the input data streams, by blocking the front front video and the front OF inputs when performing inferences. Such circumstances can occur in real world settings, e.g., the camera may be intentionally or accidentally occluded by the driver. To cope with such conditions, we fine-tune our model by performing "drop-outs" over the inputs [39]. Specifically, we block the front video and the front OF streams with probability $p = 0.5$ during the training and testing of the InterCNN with MobileNet blocks. We summarize the obtained performance in Fig. 11. Note that by blocking the front video and the front OF streams, the input of the InterCNN is the same as that fed to the two-stream CNN, while the architecture remains unchanged.

Observe that although the prediction accuracy of InterCNN drops slightly when the inputs are blocked, the occluded InterCNN remains better than the two-steam CNN in the full behavior recognition task. This suggests that out proposed architecture is highly robust to lossy input. This also confirms the effectiveness of the Interweaving modules, which we employ to improve the robustness of the model.

We conclude that, by employing MobileNet blocks in InterCNNs, we achieve the highest accuracy in the driver behavior recognition task, as compared with any of the other candidate CNN block architectures. The InterCNN + MobileNet combo also demonstrates superior computational efficiency, as it requires the lowest number of parameters, exhibits the fastest inference times and the least FLOPs. Importantly, our design is robust to lossy inputs. The sum of these advantages make the proposed InterCNN with MobileNet blocks an excellent solution for accurate driver behavior recognition, easily pluggable in modern in-vehicle intelligent systems.

F. Model Insights

Lastly, we delve into the inner workings of the InterCNN by visualizing the output of the hidden layers of the model, aiming to better understand how the neural network "thinks" of the data provided as input and what knowledge it learns.

T-distributed Stochastic Neighbor Embedding Vizualization: We first adopt the t-distributed Stochastic Neighbor Embedding (t-SNE) [40] to reduce the dimension of the last layer (the MLP layer in Fig. 5(a)), and plot the hidden

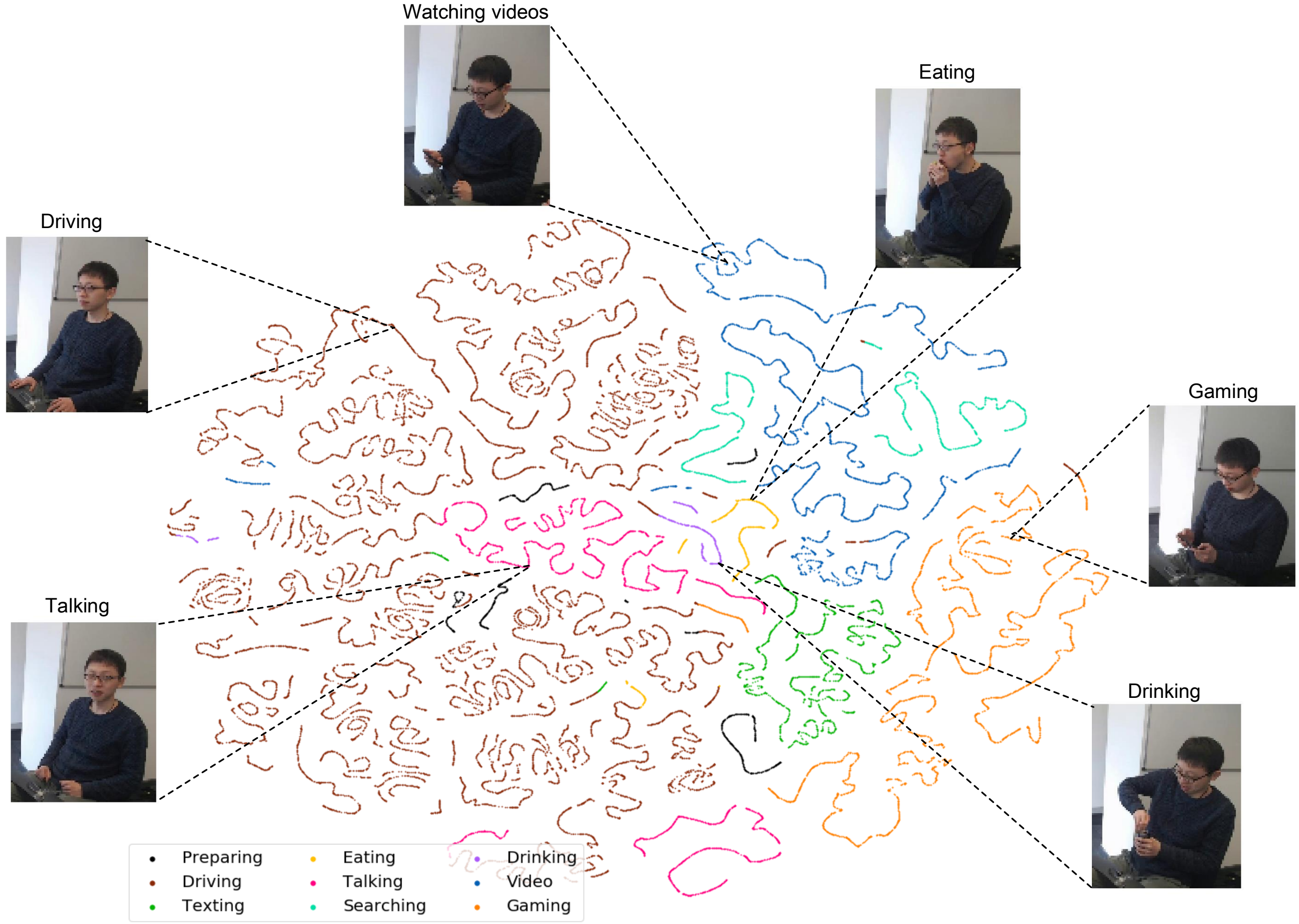


Fig. 12: Two-dimensional t-SNE embedding of the representations in the last hidden layer of the InterCNN with MobileNet blocks. Data generated using a full test video (35,711 frames).

representations of a testing video (35,711 frames) into a two-dimensional plane, as shown in Fig. 12. In general, the t-SNE approach arranges data points that have a similar code nearby in the embedding. This typically reflects how the model "thinks" of the data points, as similar data representations will be clustered together.

Interestingly, the embedding of eating and drinking remain close to each other, as both actions require to grasp a snack or drink and bring this close to the mouth. Furthermore, the embedding of actions that use a phone (i.e., web searching, texting, watching videos, and gaming) are grouped to the right side of the plane, as they are visually similar and difficult to differentiate. Moreover, as "Preparing" involves a combination of actions, including sitting down and talking to the experiment instructor, its representation appears scattered. These observations suggest that our model effectively learns the feature of different behaviors after training, as it projects similar data points onto nearby positions.

Hidden Layer Output Visualization: We also investigate the knowledge learned by the model from a different perspectives, by visualizing the output of the hidden layers of the 3D CNN block before the temporal fusion layers. This will reflect the features extracted by each individual neural network stream. We show a snapshot of such visualization in Fig. 13. Observe that the spatial streams perform "edge detection", as the object edges in the raw inputs are outlined by the 3D CNN. On the other hand, the output of the hidden layers in the temporal steams, which process the optical flows, are too abstract to interpret. In general, the abstraction level of the features extracted will increase with the depth of the architecture; it is the sum of such increasingly abstract features that enables the neural network to perform the final classification of behaviors with high accuracy.

VI. Conclusion

In this paper, we proposed an original Interwoven Convolutional Neural Network (InterCNN) to perform driver behavior recognition. Our architecture can effectively extract information from multi-stream inputs that record the activities performed by drivers from different perspectives (i.e., side video, side optical flow, front video, and front optical flow), and fuse the features extracted to perform precise classifi-

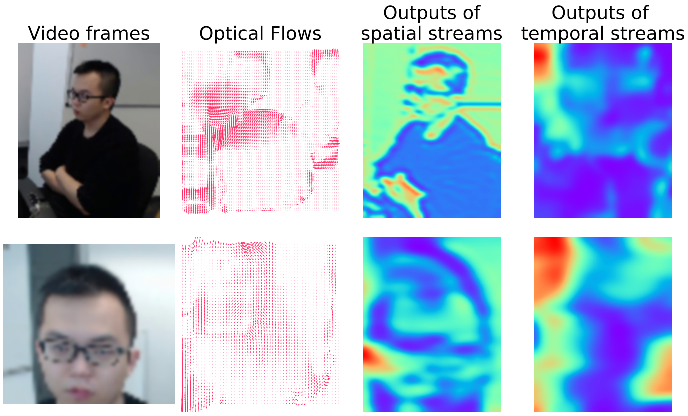


Fig. 13: The input videos frames, optical flows, and hidden outputs of the 3D CNN blocks before two temporal fusion layers. Figures shown correspond to a single instance.

cation. We further introduced a temporal voting scheme to build an ensembling system, so as to reduce the likelihood of misclassification and improve accuracy. Experiments conducted on a real-world dataset that we collected with 50 participants demonstrate that our proposal can classify 9 types of driver behaviors with 73.97% accuracy, and 5 classes of aggregated behaviors with 81.66% accuracy. Our model makes such inferences within 15 ms per instance, which satisfies the real-time constraints of modern in-vehicle systems. The proposed InterCNN is further robust to lossy data, as inference accuracy is largely preserved when the front video and front optical flow inputs are occluded.

Acknowledgment

This work was partially supported by a grant (18TLRP-B131486-02) from the Transportation and Logistics R&D Program funded by Ministry of Land, Infrastructure and Transport of the Korean government.

References

[1] Y. Liao, S. E. Li, W. Wang, Y. Wang, G. Li, and B. Cheng, “Detection of driver cognitive distraction: A comparison study of stop-controlled intersection and speed-limited highway,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1628–1637, 2016.

[2] J. C. Stutts, D. W. Reinfurt, L. Staplin, E. A. Rodgman *et al.*, “The role of driver distraction in traffic crashes,” 2001.

[3] KPMG, “Connected and Autonomous Vehicles The UK Economic Opportunity,” March 2015.

[4] SAE International, “J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” September 2016.

[5] M. Kutila, M. Jokela, G. Markkula, and M. R. Rué, “Driver distraction detection with a camera vision system,” in *Proc. IEEE International Conference on Image Processing*, vol. 6, pp. VI–201.

[6] A. S. Kulkarni and S. B. Shinde, “A review paper on monitoring driver distraction in real time using computer vision system,” in *Proc. IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, pp. 1–4.

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proc. IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[10] K. Saleh, M. Hossny, and S. Nahavandi, “Intent prediction of pedestrians via motion trajectories using stacked recurrent neural networks,” *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2018.

[11] M. Bieshaar, S. Zernetsch, A. Hubert, B. Sick, and K. Doll, “Cooperative starting movement detection of cyclists using convolutional neural networks and a boosted stacking ensemble,” *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2018.

[12] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” *arXiv preprint arXiv:1803.04311*, 2018.

[13] “The NVIDIA DRIVE™AGX takes advantage of breakthrough technologies and the power of AI to enable new levels of autonomous driving.” https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/, 2018, [Online; accessed Nov-2018].

[14] 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,” *arXiv preprint arXiv:1704.04861*, 2017.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[16] Fortune, “Driverless Cars Will Be Part of a $7 Trillion Market by 2050,” June 2017.

[17] A. Fernández, R. Usamentiaga, J. Carús, and R. Casado, “Driver distraction using visual-based sensors and algorithms,” *Sensors*, vol. 16, no. 11, p. 1805, 2016.

[18] T. Liu, Y. Yang, G.-B. Huang, Y. K. Yeo, and Z. Lin, “Driver distraction detection using semi-supervised machine learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1108–1120, 2016.

[19] A. Ragab, C. Craye, M. S. Kamel, and F. Karray, “A visual-based driver distraction recognition and detection using random forest,” in *International Conference Image Analysis and Recognition*. Springer, 2014, pp. 256–265.

[20] T. Hoang Ngan Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, “Multiple scale faster-RCNN approach to driver’s cell-phone usage and hands on steering wheel detection,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 46–53.

[21] M. S. Majdi, S. Ram, J. T. Gill, and J. J. Rodríguez, “Drive-net: Convolutional network for driver distraction detection,” in *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, 2018, pp. 1–4.

[22] D. Tran, H. M. Do, W. Sheng, H. Bai, and G. Chowdhary, “Real-time detection of distracted driving based on deep learning,” *IET Intelligent Transport Systems*, 2018.

[23] K. Yuen, S. Martin, and M. M. Trivedi, “Looking at faces in a vehicle: A deep CNN based approach and evaluation,” in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[24] C. Streiffer, R. Raghavendra, T. Benson, and M. Srivatsa, “Darnet: a deep learning solution for distracted driving detection,” in *Proceedings of ACM/IFIP/USENIX Middleware Conference*, 2017, pp. 22–28.

[25] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[26] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[27] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in neural information processing systems*, 2014, pp. 568–576.

[28] K. Schindler and L. Van Gool, “Action snippets: How many frames does human action recognition require?” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[29] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[30] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. International Conference on Machine Learning*, 2015, pp. 448–456.

[31] Klambauer, Günter and Unterthiner, Thomas and Mayr, Andreas and Hochreiter, Sepp, “Self-normalizing neural networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[33] C. Zhang and P. Patras, “Long-term mobile traffic forecasting using deep spatio-temporal neural networks,” in *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 231–240.

[34] A. Veit, M. J. Wilber, and S. Belongie, “Residual networks behave like ensembles of relatively shallow networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 550–558.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “TensorFlow: A system for large-scale machine learning.” in *OSDI*, vol. 16, 2016, pp. 265–283.

[36] H. Dong, A. Supratak, L. Mai, F. Liu, A. Oehmichen, S. Yu, and Y. Guo, “TensorLayer: A versatile library for efficient deep learning development,” in *Proc. ACM on Multimedia Conference*, 2017, pp. 1201–1204.

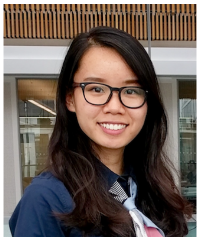
[37] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations (ICLR)*, 2015.

[38] S. F. Oberman and M. J. Flynn, “Design issues in division and other floating-point operations,” *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 154–161, 1997.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[40] L. v. d. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Chaoyun Zhang is currently working towards his Ph.D. degree at the University of Edinburgh within the School of Informatics. He obtained an MSc Artificial Intelligence from the University of Edinburgh, with a focus on machine learning. He also obtained a BSc degree from the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. His current research interests include the application of deep learning to problems in computer networking, including traffic analysis, resource allocation and network control.



Rui Li is pursuing a Ph.D. in School of Informatics at the University of Edinburgh, UK, supervised by Dr. Paul Patras. Her research interests mainly focus on applied machine learning in the networking domain. Rui obtained MSc with Distinction in Embedded System and Control Engineering from University of Leicester, UK (2014), and BEng in Communications Engineering from Northwestern Polytechnical University, China (2013).

Woojin Kim received the B.S. degree in electrical engineering from Korea Advanced Institute of Technology, Daejeon, Korea, in 2008, and the Ph.D. degrees from the Seoul National University, Seoul, Korea. He is currently a Senior Researcher at Electronics and Telecommunication Research Institute, Daejeon, Korea. His research interests are cooperative control of multiple robots or human-machine systems, and their applications.

Daesub Yoon received the M.S. and Ph.D. degree in Computer Science & Software Engineering from Auburn University. From 2001 to 2005, he was a Research Assistant with the Intelligent & Interactive System Laboratory in Auburn University. He has joined at Electronics & Telecommunication Research institute, Korea since 2005. His research interests include assistive technology, eye tracking, attentive user interface, mental workload and human factors in automated driving vehicles and smart factories.

Paul Patras [SM’18, M’11] received M.Sc. (2008) and Ph.D. (2011) degrees from Universidad Carlos III de Madrid (UC3M). He is a Lecturer and Chancellors Fellow in the School of Informatics at the University of Edinburgh, where he leads the Internet of Things Research Programme. His research interests include performance optimisation in wireless and mobile networks, applied machine learning, mobile traffic analytics, security and privacy, prototyping and test beds.