LETTER

A Deep Neural Network for Real-Time Driver Drowsiness Detection

Toan H. VU[†a)], **An DANG**[†b)], ***Nonmembers***, ***and*** **Jia-Ching WANG**[†,††c)], ***Member***

SUMMARY We develop a deep neural network (DNN) for detecting driver drowsiness in videos. The proposed DNN model that receives driver's faces extracted from video frames as inputs consists of three components - a convolutional neural network (CNN), a convolutional control gate-based recurrent neural network (ConvCGRNN), and a voting layer. The CNN is to learn facial representations from global faces which are then fed to the ConvCGRNN to learn their temporal dependencies. The voting layer works like an ensemble of many sub-classifiers to predict drowsiness state. Experimental results on the NTHU-DDD dataset show that our model not only achieve a competitive accuracy of 84.81% without any post-processing but it can work in real-time with a high speed of about 100 fps.
key words: *driver drowsiness detection, ConvCGRNN, CGRNN*

1. Introduction

Driver drowsiness detection (DDD) is one of vital components in most driver monitoring systems since drowsy driving seriously involves to many traffic accidents. Drowsiness can occur silently without self-awareness from drivers, impairs driving performance, which leads to about 6% of all crashes annually from years 2009–2013 according to [1]. Generally, an active DDD system keeping track of driver's behaviors in real-time provides timely reminders and warnings to prevent any possible accidents. Existing approaches for the DDD problem can be grouped into three major directions: systems that use driving operation information such as steering wheel angles and yaw angles [2]; systems that use physiological signals like electroencephalogram (EEG) [3]; and systems that use video signals [4], [5]. The first direction is convenient for drivers but hard to satisfy the requirement of accuracy and timeliness, while the second direction is accurate but not convenient because of wearable body sensors. In video-based DDD systems, an in-vehicle camera is adopted to capture driver's behaviors, especially facial expression, to measure drowsiness levels, which is more practically applicable. Our goal is to develop a visual DDD system that is accurate and fast to be applied in practice.

Drowsiness state can be indicated visually by some specific facial expressions such as yawning, eye-blinking, eye-closing, and nodding which can be recognized by monitoring different facial parts and global faces through time. Therefore, a general DDD system consists of two parts: a preprocessing step to extract facial information, and a classification model. The preprocessing step is to extract sequence of faces from video frames, typically involves a set of face-related techniques including face detection, face tracking, and face alignment. The way of combining these techniques affects not only accuracy but overall speed of a DDD system. In particular, previous works [4], [5], [9] require face alignment to locate and learn relevant features from specific facial regions. However, performing face detection and alignment on every frames has many challenges such as different illumination conditions, hard human poses, and occlusions. Additionally, it increases overall processing time. In our work, we work directly with global faces. Face detection and face tracking are combined together to extract driver's faces from video frames, which is simple, accurate, and very fast.

A classification model learns drowsiness-related facial features from video frames to estimate drowsiness levels. Conventionally, features are extracted spatially on each frame by a feature extraction method such as sparse coding [4], deep belief network (DBN) [5]; before being delivered to a temporal model like hidden Markov models (HMMs) to discover temporal dependencies between consecutive frames. Recently, deep learning (DL) models are widely implemented for the problem with different approaches such as long-term recurrent convolutional networks (LRCNs) [6], ensemble modeling [7], multi-task CNN [9], and 3D-CNN [8], [10]. The DL models are accurate, but computationally high in common because of window-based predictions. Furthermore, some post-processing methods [6], [9], [10] can be applied to smooth temporal predictions of model, which improves overall accuracy, but is hard to be applied in reality. In this paper, we develop a DL model that is not only accurate, but very fast at inference time. The proposed model firstly extracts facial representations from global faces by its CNN part, then its ConvCGRNN part is to learn their temporal relations while remaining spatial properties before feeding to a voting layer. Specifically, the model sequentially makes predictions by processing frame by frame instead of making window-based predictions. Thus, it is very fast, consuming much less computional cost, and capable to work in real-time.

To sumarize, our contributions are listed as follows.

Manuscript received April 28, 2019.
Manuscript revised July 15, 2019.
Manuscript publicized September 25, 2019.
[†]The authors are with the National Central University, Taoyuan, Taiwan.
[††]The author is with Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan.
a) E-mail: toanvuhong@gmail.com
b) E-mail: andtt.cit@gmail.com
c) E-mail: jcw@csie.ncu.edu.tw
DOI: 10.1587/transinf.2019EDL8079

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

– We develop a deep neural network (DNN) model for the problem of real-time driver drowsiness detection in videos. The model is designed to work directly with global faces extracted from video frames by a combination of face detection and tracking techniques.

– We propose the use of ConvCGRNN to learn temporal dependencies while still preserving spatial relations from representations extracted from global faces by a CNN, and a voting layer to measure drowsiness state.

– We conduct experiments on the public NTHU-DDD dataset to evaluate performance of the proposed model. Our model achieves a competitive accuracy of 84.81% without any post-processing, and about 100 fps inference speed.

2. Our Approach

The proposed DDD system is illustrated in Fig. 1. Given a video stream, a preprocessing step including face detection and tracking extracts driver's face from each frame. These faces are sequentially processed by a deep neural network (DNN) model to determine drowsiness state. In this section, the preprocessing step is presented before the DNN model is described in detail.

2.1 Preprocessing

The preprocessing step is to extract driver's faces from video frames. Typically, face detection is employed to extract faces from images, but it can have various difficulties such as different illumination conditions, hard poses, and occlusions. Performing face detection on every frames also increases processing time. In our work, we make a combination of face detection and tracking which is accurate and much faster. Driver's face is firstly detected by a face detector, then face locations are determined and tracked by a tracker at subsequent frames. To prevent drifting problem in tracking, we perform face detection after a cycle. If it detects successfully, we reinitialize the tracking state; otherwise, we keep tracking the current state.

In our implementation, we employ MTCNN [12] for face detection, and a correlation function in *dlib* [11] for tracking. The cycle time is fixed to 45 timesteps which corresponds to three seconds in our experiments. After extraction, faces are converted to grayscale, and resized to 128×128.

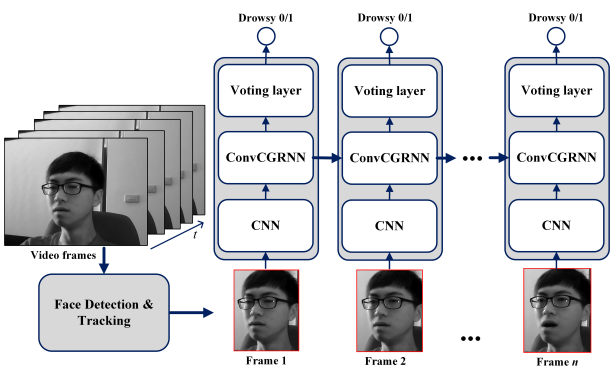

Fig. 1 The proposed driver drowsiness detection system.

2.2 Deep Neural Network Model

The proposed DNN model consists of three components: a convolutional neural network (CNN), a convolutional control gate-based recurrent neural network (ConvCGRNN), and a voting layer. It sequentially processes input faces to estimate drowsiness states (*yes* or *no*) at every timesteps.

• *CNN.* The CNN is to extract facial representations from each input face, and is constructed in a VGG-style [13] with batchnorm [15] after every convolutional layers. The details of the CNN part is shown in Table 1. Its output is a feature map with size of $(128 \times 8 \times 8)$. Each $128D$ vector in the feature map corresponds to a receptive field which covers almost the input face.

• *ConvCGRNN.* ConvCGRNN is a convolutional version of control gate-based recurrent neural network (CGRNN) [14] which is designed for effective sequence modeling with a low resource-consumption by employing an additional control gate. Figure 2 illustrates the computational structure of

Table 1 The DNN architecture. ($convX - Y$ and $convcgX - Y$ denote convolutional layer and ConvCGRNN layer respectively, where X is kernel size, Y is number of output channels. Stride size is one for all convolutional layers in both CNN and ConvCGRNN. Batchnorm and ReLU are applied after every $convX - Y$; batchnorm is applied after every convolution operators of input in ConvCGRNN, which are not shown for brevity. mpX presents a max-pooling layer where X is kernel size and stride size. $fc - X$ denotes a fully-connected layer with output size X.)

Name	Configuration	Output size
Input	*input*	$(1 \times 128 \times 128)$
CNN	$conv3 - 16 + mp2$ $conv3 - 32 + conv3 - 32 + mp2$ $conv3 - 64 + conv3 - 64 + mp2$ $conv3 - 128 + conv3 - 128 + mp2$ $conv3 - 128 + conv3 - 128$	$(128 \times 8 \times 8)$
ConvCGRNN	$convcg1 - 128$ $convcg1 - 128$	$(128 \times 8 \times 8)$
Voting	$64 \times (fc - 128) + ReLU$ $64 \times (fc - 1) + sigmoid$	(64)
Output	*global max pooling*	(1)

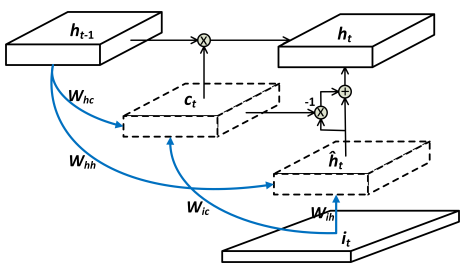

Fig. 2 Computational structure of a ConvCGRNN cell. (Blue arrows present convolution operators.)

a ConvCGRNN cell. i_t, h_t, $\hat{h}_t$, and c_t are input, hidden state, temporal state, and control gate at a time step t, respectively. They are all referred as feature maps.

At a time step t, a temporal state is given by

$$\hat{h}_t = f(W_{ih} * i_t + W_{hh} * h_{t-1}) \tag{1}$$

and a control gate is computed as

$$c_t = \sigma(W_{ic} * i_t + W_{hc} * h_{t-1}) \tag{2}$$

The new hidden state is derived as

$$h_t = c_t \otimes h_{t-1} + (1 - c_t) \otimes \hat{h}_t \tag{3}$$

where f is an activation function, and is commonly a hyperbolic tangent function; σ is a logistic sigmoid function. W_{ih}, W_{hh}, W_{ic}, and W_{hc} are weight matrices. The symbol $\otimes$ denotes element-wise multiplication, $*$ denotes convolution operator. Furthermore, batch normalization can be applied after a convolution operator of input i_t; we do not apply it after a convolution operator of the hidden state h_{t-1} to preserve gating property of ConvCGRNN.

In Eqs. (1) and (2), convolution operator is applied on both input and hidden state, which maintains spatial information of these feature maps. From Eq. (3), the temporal state $\hat{h}_t$ works like a source of new information; the previous hidden state h_{t-1} brings information from the past; and the control gate c_t scales the amount of information each source contributes to the new hidden state h_t. Thus, ConvCGRNN is able to not only learn temporal long-term dependencies, but preserve spatial properties of the input feature maps.

In this paper, we implement 2D convolution operator for ConvCGRNN structure. As shown in Table 1, the ConvCGRNN part is constructed by two ConvCGRNN layers with kernel size of 1 for all their convolution operators, and output channel size of 128. This configuration enables the ConvCGRNN to process each $128D$ vector on the CNN feature maps independently. Output of the ConvCGRNN at a timestep is a feature map with size of $(128 \times 8 \times 8)$. Each $128D$ vector on this feature map corresponds to a vector at the same location on the CNN feature map, and represents its temporal long-term dependencies.

• *Voting layer.* The voting layer is a set of two fully-connected layers without weight-sharing (Table 1). The purpose is to make independent decision for each $128D$ vector of the ConvCGRNN feature maps. We can consider each location vector on the feature maps has its own classifier. The voting layer works like an ensemble of many classifiers; the final output is a maximum value of the voting outputs.

Outputs of the model indicate the driver's state (drowsiness or non-drowsiness). Binary cross-entropy (BCE) loss is used as a loss function for the problem. Overall, the cost function is given by

$$C = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right] + \frac{\lambda}{2}\|\Theta\|^2 \tag{4}$$

Where the first term is BCE loss; the second term is weight-decay regularization term. y is the output label; $\hat{y}$ is the predicted probability. N is a batch size.

At inference time, the DNN model sequentially makes predictions by processing frame by frame without the need of resetting ConvCGRNN states. By this way, computational cost at a timestep is approximate to that of a similar structure DNN processing an input image, which is low at resource consumption.

3. Experiments

3.1 Dataset

We conduct experiments on the public NTHU-DDD dataset [5] to validate performance of the proposed model. The video dataset consists of both male and female participants with diversity in appearances, ethnicities. Videos are recorded in a simulated environment under many scenarios including *BareFace*, *Glasses*, *Sunglasses*, *Night-BareFace* and *Night-Glasses*. They capture different driver's behaviors such as *Yawning*, *Nodding*, *Looking aside*, *Talking and laughing*, *Sleepy-eyes*, *Drowsy*, and *Stillness*. Some example frames of the database are illustrated in Fig. 3. The dataset provides four frame-level annotations: *drowsiness*, *head*, *mouth*, and *eye*.

The dataset includes a training set that has 356 videos of 18 subjects, and a test set that has 20 videos of 4 subjects. In our experiments, we divide the training set into two parts: 14 subjects for training, 4 subjects for validation. Videos are resampled to 15 fps.

3.2 Training Details

We firstly initialize the CNN part by pretraining it on FER+ dataset [16] to learn rich facial representations from many subjects. Specifically, images in FER+ are resized to 64×64; two FC layers ($fc - 128 + ReLU + fc - 10$) are added right after the CNN part with an average pooling applied in the middle. This pretraining step enables the DNN model to

Fig. 3 Example frames in the NTHU-DDD dataset from different participants in various scenarios.

Table 2 Detection performances of the proposed model on the test set of the NTHU-DDD dataset.

Scenario	Non-drowsiness F1-score (%)	Drowsiness F1-score (%)	Accuracy (%)
Bareface	84.65	88.34	86.75
Glasses	70.00	82.59	77.97
Sunglasses	65.18	79.36	74.08
Night-Bareface	82.88	90.77	88.00
Night-Glasses	90.95	85.37	88.82
Overall	82.99	86.28	84.81

Table 3 Comparison of different methods on the test set of the NTHU-DDD dataset.

Method	Accuracy %
3D-DCNN [8]	71.20
Ensemble modeling [7]	73.06
Scale-Pruned 3D-CNN [10]	78.48
MSTN [6]	82.61
seqMT-DMF [9]	83.44
Human [7]	80.83
Ours	**84.81**

learn faster by a good initialization; and reduces bias in the training data which contains a small number of subjects.

For training on the target DDD problem, the preprocessing step is firstly applied to extract face sequences from videos, normalize them to grayscale with size of 128×128. To reduce overfitting, some data augmentation techniques are employed such as random rotate, random resize & crop, horizontal flip, and color jitter. We start training process for input sequences with length one (static images), then increase length of sequences to 60 (4 seconds). For optimization, we use Adam optimizer [17] with a starting learning rate of $1e-4$. Gradient clipping with the threshold of 1 is also applied. The optimal set of parameters is chosen at the highest performance on the validation set.

3.3 Experimental Results

The proposed model is evaluated on the test set of the NTHU-DDD dataset. At inference time, the model make predictions frame by frame without resetting hidden states of the ConvCGRNN part. Prediction outputs are in range of $[0, 1]$. We apply a threshold value 0.5 to decide drowsiness state.

The detection performance are shown in Table 2. The proposed system achieves an overall accuracy of 84.81%; F1-scores for detecting drowsiness and non-drowsiness are 86.28% and 82.99%, respectively. The detection performance on the *Sunglasses* and *Glasses* is the worst while it is much better on *Bareface*, *Night-Bareface*, and *Night-Glasses*. This may be due to the difference of occlusion on each scenario; and it can prove the effectiveness of using an infrared (IR) illuminator at night time in the dataset. Furthermore, we compare the results of our approach with others that make no modification on the original dataset or make no use of its test set (Table 3). Our method outperforms other methods in term of accuracy, surpasses by 3.98% human performance [7].

Table 4 Processing speeds in the proposed DDD system.

Module	Speed (fps)
Preprocessing (if face det. only)	5
Preprocessing (face det. + track)	**60**
DNN model	**100**

For comprehensive evaluation, processing time of each module in the proposed DDD system is measured. By applying face detection and tracking together, the preprocessing speed can reach 60 fps, is much faster about 12 times than that of using only face detection. The DNN model achieves a high inference speed of 100 fps. The number can be still improved in the future since our implementation in the voting layer has not been optimal because of sequential computing. Thus, the proposed system satisfies the real-time requirement of a DDD system.

4. Conclusions

We present a DDD system which is very accurate, and very fast to work in real-time. The main component of the system is a DNN model that works directly with global faces extracted from video frames by a combination of face detection and tracking techniques. The DNN consists of three component: a CNN, a ConvCGRNN, and a voting layer; works effectively to extract spatiotemporal representations from input faces, and predict drowsiness state. At inference time, the DNN sequentially processes frames from video stream without resetting ConvCGRNN states, so inference time is very fast. Experimental results on the NTHU-DDD dataset illustrate the effectiveness and efficiency of the proposed model.

Acknowledgments

This research is partially supported by the Ministry of Science and Technology under Grant Number 108-2634-F-008-004 through Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan.

References

[1] B.C. Tefft, "Prevalence of motor vehicle crashes involving drowsy drivers, United States, 2009–2013," AAA Foundation for Traffic Safety, Washington DC, 2014.

[2] Z. Li, L. Chen, J. Peng, and Y. Wu, "Automatic detection of driver fatigue using driving operation information for transportation safety," Sensors, vol.17, no.6, 1212, 2017.

[3] L.-L. Chen, Y. Zhao, J. Zhang, and J.-Z. Zou, "Automatic detection of alertness/drowsiness from physiological signals using wavelet-based nonlinear features and machine learning," Expert Syst. Appl., vol.42, no.21, pp.7344–7355, 2015.

[4] C.-Y. Chiou, W.-C. Wang, S.-C. Lu, C.-R. Huang, P.-C. Chung, and Y.-Y. Lai, "Driver monitoring using sparse representation with part-based temporal face descriptors," IEEE Trans. Intell. Transp. Syst. (Early Access), doi: 10.1109/TITS.2019.2892155.

[5] C.-H. Weng, Y.-H. Lai, and S.-H. Lai, "Driver drowsiness detection via a hierarchical temporal deep belief network," Computer Vision – ACCV 2016 Workshops, Lecture Notes in Computer Science, vol.10118, pp.117–133, Springer, Cham, 2017.

[6] T.-H. Shih and C.-T. Hsu, "MSTN: Multistage spatial-temporal network for driver drowsiness detection," Computer Vision – ACCV 2016 Workshops, Lecture Notes in Computer Science, vol.10118, pp.146–153, Springer, Cham, 2017.
[7] S. Park, F. Pan, S. Kang, and C.D. Yoo, "Driver drowsiness detection system based on feature representation learning using various deep networks," Computer Vision – ACCV 2016 Workshops, Lecture Notes in Computer Science, vol.10118, pp.154–164, Springer, Cham, 2017.
[8] J. Yu, S. Park, S. Lee, and M. Jeon, "Representation learning, scene understanding, and feature fusion for drowsiness detection," Computer Vision – ACCV 2016 Workshops, Lecture Notes in Computer Science, vol.10118, pp.165–177, Springer, Cham, 2017.
[9] L. Celona, L. Mammana, S. Bianco, and R. Schettini, "A multi-task CNN framework for driver face monitoring," 2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), pp.1–4, 2018.
[10] H. Yao, W. Zhang, R. Malhan, J. Gryak, and K. Najarian, "Filter-pruned 3D convolutional neural network for drowsiness detection," 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp.1258–1262, 2018.
[11] D.E. King, "Dlib-ml: A machine learning toolkit," Journal of Machine Learning Research, vol.10, pp.1755–1758, 2009.
[12] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," IEEE Signal Process. Lett., vol.23, no.10, pp.1499–1503, 2016.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR 2015.
[14] T.H. Vu, L. Dung, and J.-C. Wang, "Transportation mode detection on mobile devices using recurrent nets," Proc. 24th ACM International Conference on Multimedia, MM '16, pp.392–396, 2016.
[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Proc. 32nd International Conference on Machine Learning, PMLR 37, pp.448–456, 2015.
[16] E. Barsoum, C. Zhang, C.C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," Proc. 18th ACM International Conference on Multimodal Interaction, ICMI 2016, pp.279–283, 2016.
[17] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR 2015.