IET Computer Vision

Research Article

IET Journals
The Institution of Engineering and Technology

Driving posture recognition by convolutional neural networks

ISSN 1751-9632
Received on 20th May 2015
Revised on 12th August 2015
Accepted on 1st September 2015
doi: 10.1049/iet-cvi.2015.0175
www.ietdl.org

Chao Yan[1,2] ✉, *Frans Coenen*[2], *Bailing Zhang*[1]

[1]*Department of Computer Science and Software Engineering, Xian Jiaotong-Liverpool University, SIP, Suzhou 215123, People's Republic of China*
[2]*Department of Computer Science, The University of Liverpool, Liverpool L69 3BX, UK*
✉ *E-mail: choise.yan@163.com*

Abstract: Driver fatigue and inattention have long been recognised as the main contributing factors in traffic accidents. This study presents a novel system which applies convolutional neural network (CNN) to automatically learn and predict pre-defined driving postures. The main idea is to monitor driver hand position with discriminative information extracted to predict safe/unsafe driving posture. In comparison to previous approaches, CNNs can automatically learn discriminative features directly from raw images. In the authors' works, a CNN model was first pre-trained by an unsupervised feature learning method called sparse filtering, and subsequently fine-tuned with classification. The approach was verified using the Southeast University driving posture dataset, which comprised of video clips covering four driving postures, including normal driving, responding to a cell phone call, eating, and smoking. Compared with other popular approaches with different image descriptors and classification methods, the authors' scheme achieves the best performance with an overall accuracy of 99.78%. To evaluate the effectiveness and generalisation performance in more realistic conditions, the method was further tested using other two specially designed datasets which takes into account of the poor illuminations and different road conditions, achieving an overall accuracy of 99.3 and 95.77%, respectively.

1 Introduction

With the ever-growing traffic density, the number of road accidents is anticipated to further increase. Unsafe and dangerous driving accounts for the death of more than one million lives and over 50 million serious injuries worldwide each year [1]. Finding solutions to reduce road accidents and to improve traffic safety has become a top priority for many government agencies and automobile manufactures alike. It has become imperative to the development of intelligent driver assistance systems (IDAS) which is able to continuously monitor, not just the surrounding environment and vehicle state, but also driver behaviours.

Previous works using IDAS to prevent traffic accident can be categorised into two main streams of activities, which are the vehicle-oriented approaches [2, 3] and the driver-oriented approaches [4–7]. For the first one, driver vigilance is analysed through vehicle behaviour including movement of steering wheel, pressure on the acceleration pedal, speed, deviations from lane position, response time against an obstacle braking, and so on. The main limitations of these approaches [2, 3] include their dependence on the shape of the road, the vehicle performance, and the manner of driving. For the second approach, driver behaviour monitoring is based on the analysis of physiological and biomedical signals such as heart rate, brain activity, temperature, vascular activity, and muscular activity. Such methods [4] rely on wearable sensors which decrease user experience and increase hardware cost. An alternative way to analyse driver behaviour is using a camera. There are three categories of vision-based approaches to automatically monitor the unsafe driver behaviour: (i) gaze and head pose analysis for the prediction of driver behaviour and intention [5, 8–11], (ii) extraction of fatigue cues from driver facial image [6, 12–14], and (iii) characterisation (in the context of safe versus unsafe driving behaviour) of driver body postures, including the positioning of arms, hands, and feet [7, 15–17]. Despite the encouraging performances under appropriate conditions, the proposed approaches share a common disadvantage of being *ad hoc*. Most of the vision-based methods follow a two-step framework: (i) extraction of hand-crafted features from raw data, usually with certain assumptions about the circumstances under which the data was taken, and (ii) learning classifiers based on the obtained feature. Methods under such a framework cannot reach an optimal balance between the discriminability of the extracted features and the robustness of the chosen classifier. The reason is the uncertainty of what features are important for the task at hand since the choice of features is highly problem dependent in real-world scenarios.

Recently, there has been growing interest in the development of deep learning models for various vision tasks [18–20]. Deep learning models, generally features of learning multiple layers of feature hierarchies, with increasingly abstract representations extracted at each stage. Such learning machines can be trained using either supervised or unsupervised approaches, and the resulting systems have been shown to yield competitive performance in speech recognition [21, 22], natural language processing [23], image classification [24–27], visual object detection [28], and other visual tasks [29–33].

One of the most successful deep learning models is the convolutional neural network (CNN) [20, 24], a hierarchical multilayered neural network able to learn visual patterns directly from the image pixels. In CNNs, small patches of the image (dubbed as a local receptive field) are inputted to the first layer of the hierarchical structure. Information generally passes on the different layers of the network, and at each layer trainable filters and local neighbourhood pooling operations are exploited in order to produce salient features for the data observed. In addition, the method provides a level of invariance to shift, scale, and rotation as the local receptive field allows the processing unit access to elementary features such as oriented edges or corners. It has been repeatedly proved that CNN is powerful to learn rich features from the training set automatically.

In this paper, we apply CNN architecture to represent and recognise driving postures, which aims at building high-level

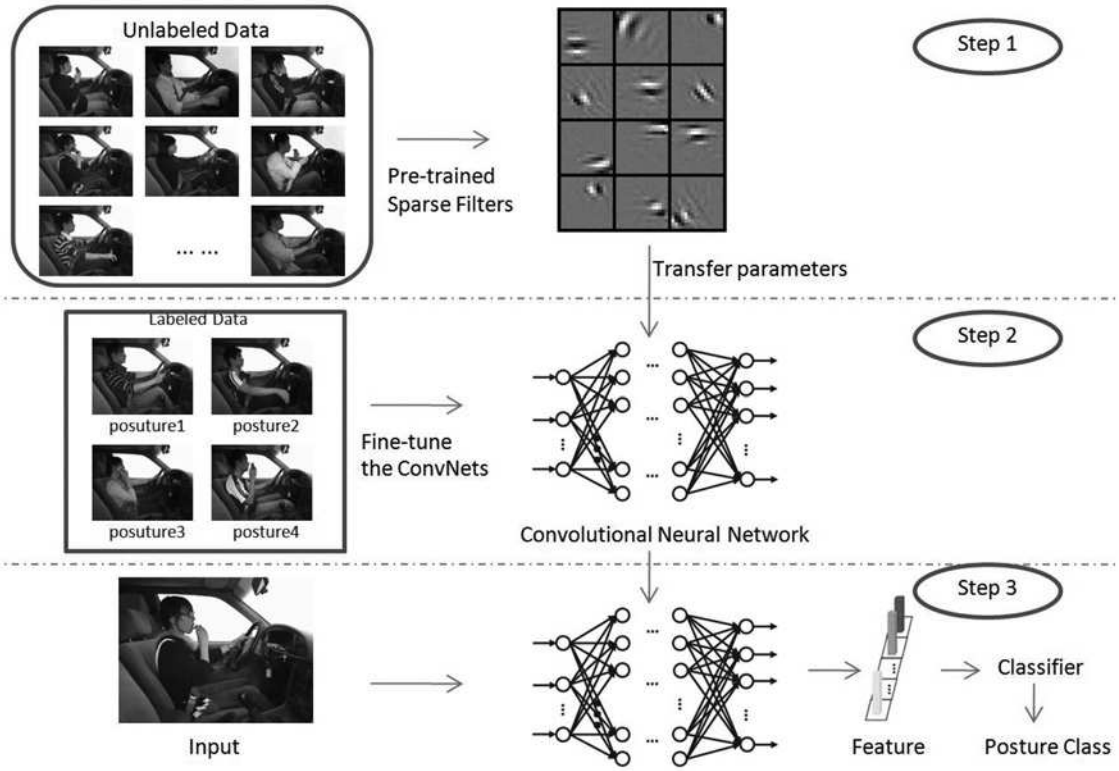

Fig. 1 *Frameworks of our method*

feature representation from low-level input automatically with minimal domain knowledge of the problem. Our work focus on the characterisation of driving posture, with high-level features extracted hierarchically from raw input image. Each convolutional layer generates feature maps using sliding filters on a local receptive field in the maps of the preceding layer (input or max-pooling layer). The map sizes decrease layer by layer such that the extracted feature becomes more complex and global. Then, the output is inputted to a fully connected multilayer perceptron (MLP) classifier. The proposed approach was evaluated on the Southeast University (SEU) driving posture dataset [13], demonstrating competitive performance.

The key contributions of this work can be summarised as follows:

(i) To recognise driving posture, this paper proposed to build a deep CNN in which trainable filters and local neighbourhood pooling operations are applied alternatively to automatically explore salient features. Using CNN to learn rich features from the training set is more generic and requires minimal domain knowledge of the problem compared with hand crafted feature in previous approaches.
(ii) We used sparse filter [34] to pre-train the filters in our networks, with advantages including (i) acceleration of training for faster convergence, and (ii) a better generalisation performance. In addition, we setup experiment to evaluate the CNN architecture selection, with max-pooling and rectifier linear unit (ReLU) identified as better options for pooling operation and activation function, respectively.
(iii) The proposed approach was evaluated on the SEU driving posture dataset [13]. To account for the poor illuminations and different road conditions, we created two sets of video data, namely, the Driving-Posture-atNight dataset and the Driving-Posture-inReal dataset. The performances achieved on these three datasets are characterised by accuracies of 99.47, 99.3, and 95.77%, respectively.

The rest of the paper is organised as follows. Section 2 presents an overview of our proposed method and the SEU driving posture dataset, while Section 3 gives a detailed introduction to the CNN followed by the training details in Section 4. Section 5 reports experiment results for the performance evaluation, followed by some conclusions presented in Section 6.

2 System overview

Following the approach proposed by Zhao *et al.* [13], our work also focus on the characterisation of driving postures with reference to driver hand position. However, hand region is difficult to be accurately estimated due to the limitations of skin region segmentation algorithms and the variation of light condition. In our approach, we use the whole frame extracted from raw video instead of hand region as input. This is achieved by using CNN which can automatically learn discriminative feature representation directly from raw data.

To have an overview of the proposed approach, Fig. 1 gives an illustration of the driving posture recognition system. The system comprises three steps: (i) unsupervised pre-training of the network with unlabelled data, (ii) fine-tuning the network with four classes of the labelled data, and (iii) feature extraction using the network from input for classification.

2.1 SEU driving posture dataset

To test the proposed driving posture recognition approach, the SEU driving posture dataset was used. This data was first created by Zhao *et al.* [13]. Each video included in the dataset was obtained using a side-mounted Logitech C905 CCD camera under day lighting conditions with a resolution of 640×480. Ten male drivers and ten female drivers participated in the creation of the dataset. Each video was recorded under normal day light conditions. We extracted all frames in these videos and manually labelled four pre-defined postures including:

(i) normal driving (posture 1),
(ii) operating the shift gear (posture 2),

IET Comput. Vis., 2016, Vol. 10, Iss. 2, pp. 103–114

Fig. 2 *Example images from the SEU driving dataset [13]. First column is normal driving posture; second column is the posture of operating the shift gear; third column is the posture of eating or smoking; forth column is the posture of responding to a cell phone*

(iii) eating and smoking (posture 3),
(iv) responding to a cell phone (posture 4).

Some selected samples are shown in Fig. 2. Each posture from (i) to (iv) contains 46 081, 12 000, 18 181, and 16 211 samples, respectively.

2.2 New driving posture dataset

To further evaluate the effectiveness and generalisation performance of our approach, we built two different datasets, namely, Driving-Posture-atNight and Driving-Posture-inReal.

The first dataset, Driving-Posture-atNight, was recorded using a UWISH UC-H7225 infrared camera at night under low illumination conditions. Similar to SEU dataset, ten male and ten female participants performed the same four pre-defined postures, namely normal driving, operating the shift gear, eating/smoking, and responding to a cell phone. All frames were extracted and manually labelled. Some examples are shown in Fig. 3. They are divided into three subsets, that is, training set, validation set, and test set, with 24 210, 1000, and 4200 samples, respectively.

To further evaluate the system performance in more realistic condition, the second dataset, Driving-Posture-inReal, was recorded using a Philips CVR300 car driving recorder. It was side mounted in front window of a family car. In the dataset creation, five experienced drivers drove the car in turn on city road. The drivers were required to conduct two activities while driving, that is, eating cookies and responding to cellphone calls. All frames were extracted and manually labelled into four classes as previous. Some examples are shown in Fig. 4. They are divided into three subsets, that is, training set, validation set, and test set, with 14 230, 1000, and 2500 samples, respectively.

3 Deep CNN architecture

In this section, we will briefly overview the CNN architecture, with appropriate explanation of each building block.

3.1 Overall network architecture

The overall CNN architecture is shown in Fig. 5. The network consists of three convolution stages followed by three fully connected layers. Each convolution stage includes convolutional layer, non-linear activation layer, local response normalisation layers, and max-pooling layer. The non-linear activation layer and local response normalisation layers were not illustrated as data size was not changed in these two layers. Using shorthand notation, the full architecture can be denoted as $C(12,5,1)$-$\tilde{A}$-N-P-$C(16,5,1)$-$\tilde{A}$-N-P-$C(20,4,1)$-$\tilde{A}$-N-P-FC(512)-$\tilde{A}$-FC(128)-$\tilde{A}$-FC(4)-$\tilde{A}$, where $C(d, f, s)$ indicates a convolutional layer with d filters of spatial size $f \times f$, applied to the input with stride s. $\tilde{A}$ is the non-linear activation function, which uses ReLU activation function [35]. FC(n) is a fully connected layer with n output nodes. All pooling layers P use max-pooling in non-overlapping 2×2 regions and all normalisation layers N are defined as described by Krizhevsky *et al.* in [24] and use the same parameters: $k = 2$, $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.5$. The final layer is connected to a softmax layer with dense connections. The structure of the networks and the hyper-parameters were empirically initialised based on the previous work [36] using

Fig. 3 *Example images from the Driving-Posture-atNight dataset. Column 1: normal driving; column 2: operating the shift gear; column 3: eating or smoking; column 4: responding to a cell phone*

ConvNets. A cross-validation experiment was conducted to optimise the selection of network architecture, which will be elaborated in Section 5.2.

3.2 Convolution layer

The CNN is a biologically inspired variant of the MLP, also known as shared weight neural networks introduced by LeCun *et al.* [20]. From Hubel and Wiesels [37] early works on the cats visual cortex, it is known that the visual cortex contains a complex arrangement of cells. These cells are sensitive to small sub-regions of the visual field, called a receptive field. The sub-regions are tiled to cover the entire visual field. These cells act as local filters over the input space and are well suited to exploit the strong spatially local correlation present in natural images.

In the CNN architecture, the local receptive field (kernel or filter) is replicated across the entire visual field to form a feature map, which is known as convolution operation. The convolution operations share the same parameterisation (weight vector and bias). Such a sharing of weights reduces the number of free variables, while increasing the generalisation performance of the network. Weights (kernels or filters) are initialised as random and will be learned to be edge, colour, or specific patterns detectors.

The convolution operation is expressed as

$$y^j = f_{\text{acti}}\left(b^j + \sum_i \boldsymbol{w}^{ij} * x^i\right) \tag{1}$$

where x^i and y^j are the ith input feature map and the jth output feature map, respectively. $\boldsymbol{w}^{ij}$ is the weights of the convolution filter.

IET Comput. Vis., 2016, Vol. 10, Iss. 2, pp. 103–114

Fig. 4 *Example images from the Driving-Posture-inReal dataset. Column 1: normal driving; column 2: operating the shift gear; column 3: eating or smoking; column 4: responding to a cell phone*

* denotes the convolution operation. b^j and $f_{\text{acti}}(\cdot)$ is the bias and activation function of the jth output feature map, respectively. The non-linear activation function $f_{\text{acti}}(\cdot)$ is regarded as a single layer as opposite to a function intergraded in convolution layer, and will be introduced in more details in the following.

3.3 Non-linear activation layer

The convolution layer performs as a linear filter. To form a non-linear complex model, non-linear activation functions are needed to be passed, which transforms the input value non-linearly to the output value of the neuron. Originally, sigmoidal activation functions were often used as the activation function in neural networks, for example, the sigmoid and hyperbolic tangent functions

$$\text{sigmoid}(x) = \frac{1}{1 + \text{e}^{-x}} \tag{2}$$

$$\tanh(x) = \frac{\text{e}^{2x} - 1}{\text{e}^{2x} + 1} \tag{3}$$

Sigmoidal functions are bounded by minimum and maximum values as illustrated in Fig. 6*a*. The sigmoidal function has the so-called saturation problem. In a saturated neuron, the gradient of the activation function approximates zero, which will diminish the gradient flow [38] to the lower layers in the neural network. This causes either the error rate decays or explodes exponentially to the lower layers, which makes the training very slow.

Glorot *et al.* [35] proposed an alternative activation function, called ReLU, as in (4), and compared it with a softplus activation function [39], a smooth version of the ReLU, as in (5)

$$\text{ReLU}(x) = \max(x, 0) \tag{4}$$

$$\text{softplus}(x) = \log(1 + \text{e}^{x}) \tag{5}$$

These two less saturated activation functions can be illustrated in Fig. 6*b*. Recent CNN-based approaches [24–27, 32, 33] applied the ReLU as the non-linear activation function for both convolution layer and full connection layer, generally with faster training speed as reported by Krizhevsky *et al.* [24].

3.4 Pooling layer

An output map with local feature is extracted by previous linear convolution, adding bias and non-linear activation. Due to the replication of weights in a convolutional layer, the detected features may be still sensitive to the precise positions of the input pattern, which is harmful to the performance if it is followed by subsequent classification. To solve the problem, a reasonable approach is to pool the features, which has three major advantages: (i) pooling in an efficient form of dimensionality reduction, which decreases feature maps resolution and reduces computation for upper layers in CNN. It throws away unnecessary information and only preserves the most critical information [40], (ii) pooling makes activations in a neural network less sensitive to the specific structure of the neural network [41]. Moreover, it makes a network less sensitive to the exact location of the pixels, which results in a form of translation invariance, and (iii) pooling summarises the output of multiple neurons from convolutional layers with the essence of taking nearby feature detectors and forming local or global bag of features [40].

Typical pooling functions are average and maximum, generally with the name of average-pooling (subsampling, downsampling, and meanpooling) and max-pooling layers, as in (6) and (7), respectively

$$y^i_{m,n} = \frac{1}{s^2} \sum_{0 \le \lambda, \mu < s} x^i_{m \cdot s + \lambda, n \cdot s + \mu} \tag{6}$$

$$y^i_{m,n} = \max_{0 \le \lambda, \mu < s} x^i_{m \cdot s + \lambda, n \cdot s + \mu} \tag{7}$$

where each neuron in the ith output map y^i pools over an $s \times s$ non-overlapping local region in the ith input map x^i. Average-pooling as the name suggest basically takes the arithmetic mean of the elements in each pooling region while max-pooling selects the largest element form the input.

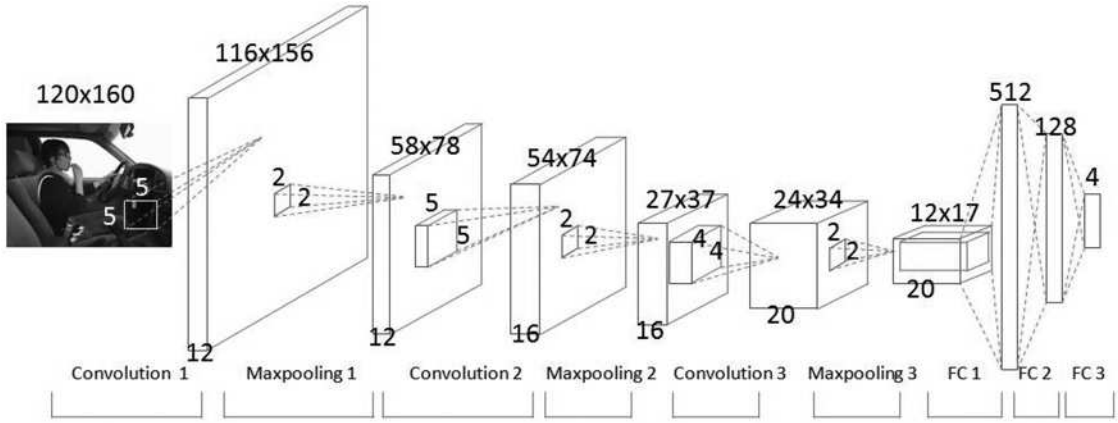

Fig. 5 *Architecture of our unsupervised CNN. Network contains three stages, each of which consisted of convolution layer, non-linear activation layer, local response normalisation layer, and max-pooling layer. Only convolution and max-pooling layers which change the data size during operating are illustrated here*

3.5 Local normalisation layer

There are two streams of normalisation techniques being used in CNN architectures, including local contrast normalisation [42, 43] and local response normalisation [24].

The local contrast normalisation is to eliminate the higher-order statistical dependencies in photographic images [44, 45], which is especially important in un-constrained environment machine vision tasks. Inspired by computational neuroscience models [46, 47], the local contrast normalisation method was proposed as a layer in CNN architecture [42, 43]. Alternatively, Krizhevsky *et al.* [24] founds that their normalisation method, local response normalisation, increases generalisation and decreases error rates in the ImageNet classification experiment. Denoting by $x^i_{m,n}$ the ith input activity of a neuron after applying non-linearity and pooling, the response-normalised activity $y^i_{m,n}$ is given by the following equation

$$y^i_{m,n} = x^i_{m,n} \Big/ \left[k + \alpha \sum_{j=\max(0,i-l/2)}^{\min(L-1,i+l/2)} (x^j_{m,n})^2 \right]^{\beta} \tag{8}$$

where the sum $\sum$ runs over l adjacent kernel maps at the same spatial position, and L is the total number of kernels in the layer. The constants k, l, α, and β are hyper-parameters whose values are determined using a validation set. The ordering of the kernel maps is arbitrary and determined before training begins. This sort of response normalisation implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs computed using different kernels. This scheme bears some resemblance to the local contrast normalisation scheme, but would be more appropriately termed as 'brightness normalisation', since no subtraction of the mean activity is involved. For the impressive performance of ImageNet benchmark classification [24] and for fair comparison, the local response normalisation is frequently used in recent CNN architecture [24–27, 32, 33]. Hence, we adopted a same normalisation as in (8), with the same parameters as in [24].

3.6 Full connection layer

The four layers, convolution layer, non-linear layer, pooling layer, and normalisation layer, are combined hierarchically to form a convolution stage (block). Generally, the raw input image will be passed through several convolution stages for extracting complex descriptive features in conventional CNN architecture. In the output of topmost convolution stage, all small-size feature maps are concatenated into a long vector. Such a vector plays the same role as hand-coded features and it is fed to a full connection layer. A standard full connection operation follows the conventional

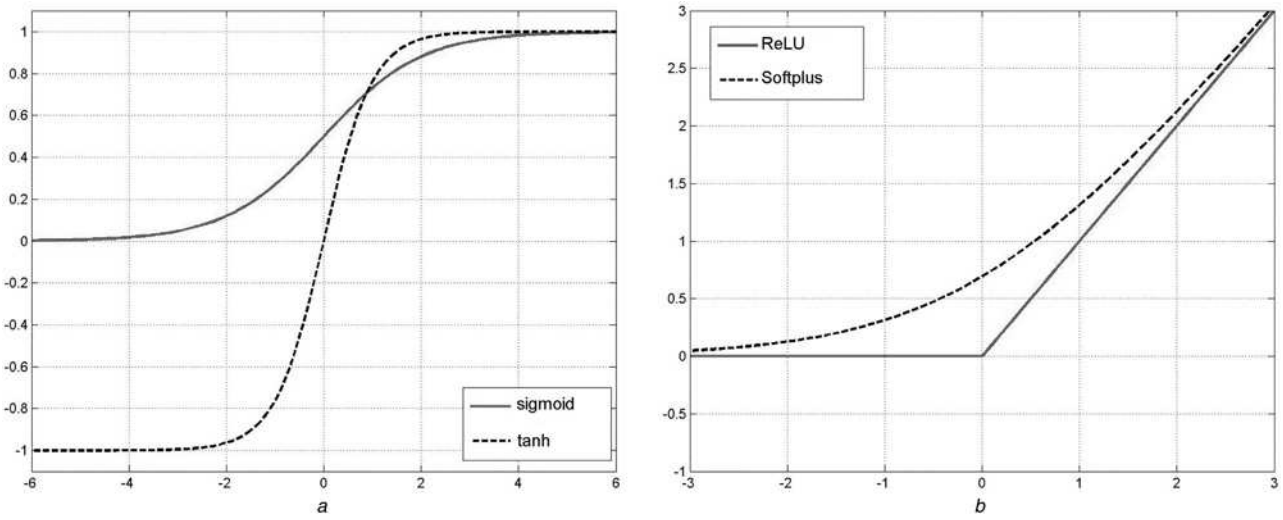

Fig. 6 *Plots of four activation functions*

a Sigmoid against Tanh
b ReLU against Softplus

IET Comput. Vis., 2016, Vol. 10, Iss. 2, pp. 103–114

MLP, which can be expressed as

$$y^j = f_{\text{acti}}\left(\sum_{i=1}^{m} \boldsymbol{w}^{ij} x^i + b\right) \tag{9}$$

where x^i is the ith neuron of a m-dimension input vector, y^j is the jth neuron of a n-dimension output vector, $\boldsymbol{w}$ is a $m \times n$ weight matrix, b is called bias units which correspond to the intercept term, and $f_{\text{acti}}(\cdot)$ is the activation function as introduced in Section 3.3.

3.7 Output layer

As the fully connected layers receive feature vector from the topmost convolution stage, the output layer can generate a probability distribution over the output classes. Toward the purpose, the output of the last fully connected layer is fed to a K-way softmax (where K is the number of classes) layer, which is same as a multi-class logistic regression. If we denote by x^i the ith input to the output layer, then the probability of the ith class, p^i, is calculated by the following softmax function

$$p^i = \exp(x^i) / \sum_{j=1}^{K} \exp(x^j) \tag{10}$$

4 Training procedure

In this section, we first briefly outline the back-propagation algorithm for the CNN training. Then the practically important issue, that is, the pre-training, will be subsequently discussed.

4.1 Learning through back propagation

In CNN training, the famous back-propagation algorithm [48] is often used to propagate errors in the network architecture. Training is composed of two steps: (i) feed forward the training data through the network till the final output layer, and finally calculate the error or a loss function value, and (ii) back propagate the error/loss layer by layer from top to bottom, and update the weights in respective layers based on the back-propagated errors.

An appropriate loss estimation for the output from the final layer is the cross-entropy loss [49], which has faster training process than the conventional mean square error. In the output layer, the cross-entropy loss function is given by

$$L = -\sum_{j=1}^{K} [t^j \log(p^j) + (1 - t^j) \log(1 - p^j)] \tag{11}$$

where x^j and p^j are the jth input and output, respectively, as defined in Section 3.7, K is the number of output neurons (class), and t^j is the jth class's one-per-class encoded target label.

When performing back propagation, the first step is to calculate the loss gradient by partial derivative as follows

$$\begin{aligned} \delta^j &= \frac{\partial L^j}{\partial p^j} \cdot \frac{\partial p^j}{\partial x^j} = \left(-\frac{t^j}{p^j} + \frac{1 - t^j}{1 - p^j}\right) \cdot p^j(1 - p^j) \\ &= \frac{p^j - t^j}{p^j(1 - p^j)} \cdot p^j(1 - p^j) \\ &= p^j - t^j \end{aligned} \tag{12}$$

where δ^j is the loss gradient in the output layer, which will be back propagated to the topmost full connection layer as an error.

In an lth full connection layer, an error δ^j_{l+1}, is back propagated from an upper $(l+1)$th layer, then the error δ^i_l and the weight gradient $\Delta \boldsymbol{w}^{ij}_l$ in this layer is given by (13) and (14), respectively

$$\delta^i_l = \sum_{j=1}^{n} \boldsymbol{w}^{ij}_l \cdot \Delta f_{\text{acti}}(\cdot)_l \cdot \delta^j_{l+1} \tag{13}$$

$$\Delta \boldsymbol{w}^{ij}_l = x^i_l \cdot \Delta f_{\text{acti}}(\cdot)_l \cdot \delta^j_{l+1} \tag{14}$$

where $\boldsymbol{w}^{ij}$ is the $m \times n$ weight matrix, n and x^i_l are the total number of the output neuron and input neuron when feed forwarding, respectively. $\Delta f_{\text{acti}}(\cdot)_l$ is the gradients of the non-linearity activation function, the error δ^i_l will be back propagated to the lower $(l-1)$th layer. More details about the CNN training procedure can be referred to [50].

4.2 Pre-training

CNN architecture strongly depends on large amounts of training data for good generalisation. When the amount of labelled training data is limited, directly training a high-capacity CNN from only a few thousand training images is problematic. Researches [51] have shown an alternative solution to alleviate the CNN requirement, that is, choosing an optimised starting point which can be pre-trained by transferring parameters replacement of random initialisation.

The first layer of many deep neural networks trained on natural images learns features similar to Gabor filters and colour blobs. It seems that the first-layer features appear not to be dependent on a particular dataset or task, but general in that they are applicable to common visual datasets and tasks. Therefore, an intuitive hypothesis was proposed in [52] that features must eventually be transformed from general to specific layers by layers in a conventional deep neural network, which may provide the theoretical support for the pre-training by transferring parameters.

Following the guideline of pre-training methodology as discussed above, we introduce the sparse filtering [34] to learn the filter in each convolution layer of our network. Sparse filtering is an unsupervised feature learning algorithm which optimises for the sparsity in the feature distribution and avoids explicit modelling of the data

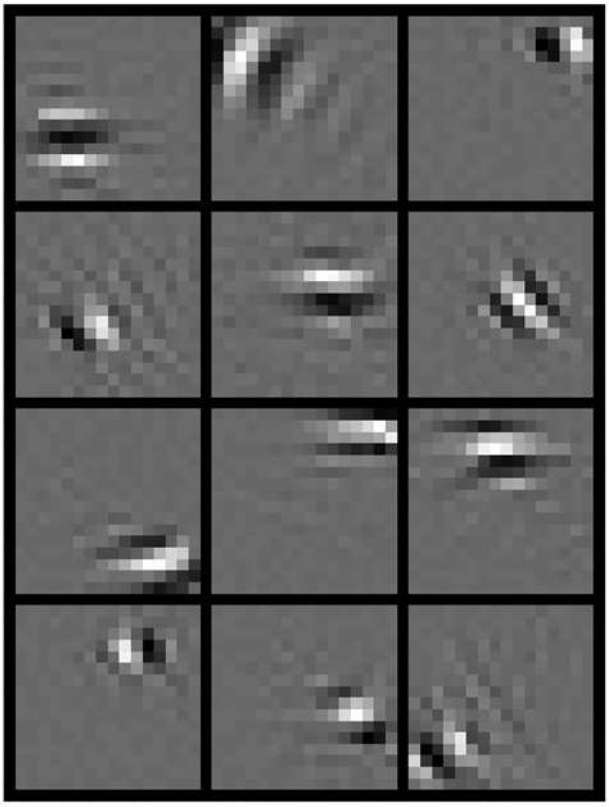

Fig. 7 *Unsupervised pre-trained sparse filters of the first convolution layer*

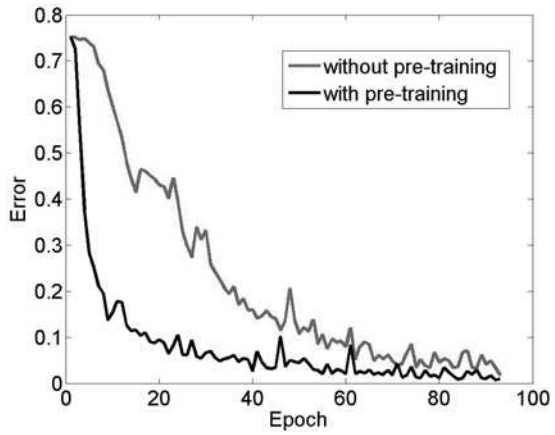

Fig. 8 *Improvement by using pre-train*

distribution. Compared with other unsupervised techniques such as autoencoder [53], sparse coding [54], and sparse restricted boltzmann machines (RBMs) [18, 55], sparse filtering is easy to implement and hyperparameter free. In experiment, we randomly extracted patches with size 5×5 from the video dataset and the sparse filtering method is utilised to learn the filters. The learnt filters of the first layer are shown in Fig. 7. As we expected, the filters mainly preserve the edge, point, and junction information of the driver. Fig. 8 shows a testing error versus epoch, illustrating the effect of using pre-train. From the figure, it is clear that pre-training improves convergence speed about 20 epoches.

5 Experiment

In this section, we first introduce some implementation details in Section 5.1. Then three evaluation experiments to select the CNN structure, activation function, pooling method, and other hyper-parameters are presented in Section 5.2. The CNN model is applied to verify the effectiveness of the proposed algorithm on the SEU driving posture database and our own driving posture database as described in Sections 5.3 and 5.4. Finally, some published approaches with hand-coded features are compared in Section 5.6.

The SEU driving posture dataset contains 20 video clips. All experiments are conducted using five-fold cross-validation. The original 20 videos are randomly divided into five folds, each containing four video clips. In five-fold cross-validation, one of the folds is retained for testing while the remaining four folds for training. The cross-validation process will then be repeated five times, which means that each of the fold will be used exactly once as the testing data. In each training, 5% of the training data are randomly selected from each driver as validation data. The selection of validation data is consistent with dataset distribution.

5.1 Implementation detail

We train our models using stochastic gradient decent with a batch size of 128 examples, momentum of 0.6, and weight decay of 0.0005. The learning rate is initialised as 0.01 for all trainable layers and adapted during training. How to adaptively control the learning rate within a reasonable range is an important issue in CNN learning. A too small learning rate makes the convergence rather slow, while a too big learning rate would make the network parameters vibrated. We proposed an adaptive learning rate by monitoring the loss function value and the validation error. We divide the learning rate by 2 if the loss value and validation error stops decreasing. To further prevent possible overfitting, we apply dropout and data augmentation as performed in [24].

The experiments were implemented on our graphics processing unit (GPU) CNN package written in C++ language based on NVIDIA CUDA and cuDNNv2. Our experiments are conducted on a NVIDIA GTX Titan GPU and a 4-core Intel(R) Core i7-3770 3.40 GHz computer.

5.2 Architecture selection

We conducted three experiments to select the network architectures, including convolution layer capacity, non-linear activation function, and pooling method. Initially, a default architecture was chosen, denoted as $C(9,5,1)$-$\tilde{A}$-N-P-$C(12,5,1)$-$\tilde{A}$-N-P-$C(15,4,1)$-$\tilde{A}$-N-P-FC(512)-$\tilde{A}$-FC(128)-$\tilde{A}$-FC(4)-$\tilde{A}$, with the notation meaning explained in Section 3.1. Then cross-validation was applied to optimise all the hyperparameters one by one.

5.2.1 Architecture capacity: According to learning theory, if the architecture has too much capacity, it tends to overfit the training data with poor generalisation. If the model has too little capacity, it underfits the training data, and both the training error and the test error are high. The capacity here means (i) the depth of the deep CNNs, and generally stands for the number of the repeated convolution stages, and (ii) the width of the network, which means the filter numbers in each convolutional layer.

The depth mainly depends on the size of raw input image and the complexity of the task. Compared with those 10,000 face image classification tasks [33], the complexity of our work is relatively lower. The original image is with size 480×640. To save computation resources, we downsized the images to 120×160, without losing discriminative information to classify the postures. We selected the filter size of 5×5 for first and second convolution layers, and the filter size of 4×4 for the third convolution layer. Each subsequent pooling layer uses a non-overlapping 2×2 kernels which is same as most CNN-based approaches. Therefore, the size of the output feature map is 12×17 after three stages of convolution as shown in Fig. 5. The selection is based on our empirical finding that the resolution of feature map is small enough to terminate convolution.

With regard to the width of the network, we empirically set the filter size as 9, 12, and 15 in the three convolution layers, respectively. The subsequent pooling layers will decrease resolution of the feature maps. To prevent the information from being lost too quickly, the filter size will generally increase. Here, we setup experiment to compare with three other groups of filter number, that is, 6–8–10, 12–16–20 and 18–24–30, where the notation of x–y–z means x number filters, y number filters, and z number filters in the first, second, and third convolution layers, respectively. The validation error and testing error with epoch are shown in Figs. 9*a* and *b*, respectively. From the figure, the network comparing four combinations with different number of filters converges since 10 epoches and is stable after 20 epoches. However, the black dash line which stands for the filter number group of 12–16–20, exhibits a minor early convergence compared with other three groups. Hence, we choose this group as our filter numbers in each convolution layer.

5.2.2 Non-linear activation function and pooling method: In this section, we evaluate the non-linear activation function and pooling method in the CNN network. The ReLU has been reported faster convergence than conventional sigmoidal functions, and we re-implemented these activation functions with the validation error with epoch illustrated in Fig. 10*a*. In the figure, we plot the first ten epoches, which demonstrates a higher training speed of ReLU and softplus. ReLU has been reported with lower test error than softplus in [35], which, however, has not been observed in our experiments.

Typical pooling methods include average-pooling and max-pooling. Average-pooling method pools out the average value within a given reception field, while max-pooling method pools out the max value. The validation errors with epoch for these two pooling methods are plotted in Fig. 10*b*. It is obvious that max-pooling yields a better performance.

IET Comput. Vis., 2016, Vol. 10, Iss. 2, pp. 103–114

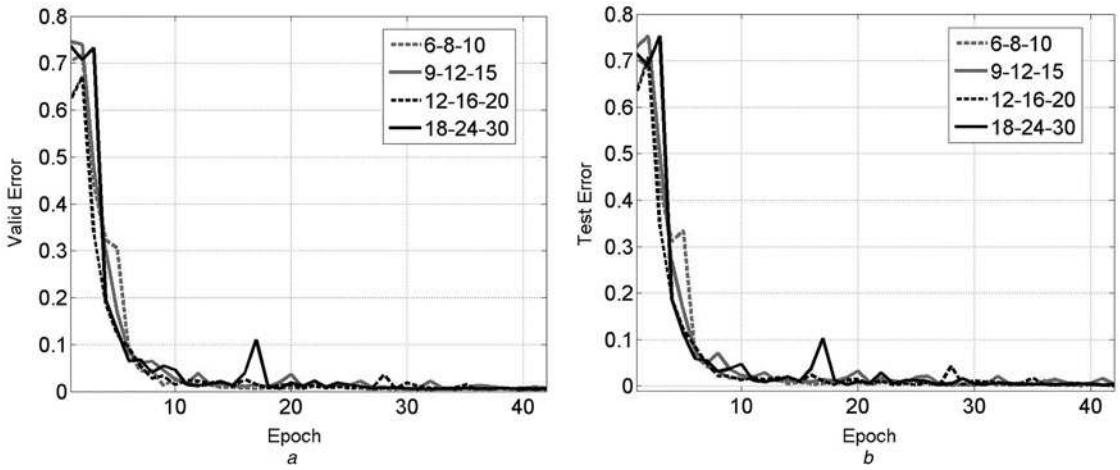

Fig. 9 *Selecting filter numbers in each convolution layer*

a Validation error
b Testing error

5.3 Evaluation with SEU driving posture database

With the selected architecture described above, we carried out experiments with CNN for training and testing using the SEU dataset. The experiment procedure is five-fold cross-validation. The first time result is illustrated in Fig. 11. In Fig. 11*a*, the solid line shows the global loss, the dash line is the moving average of the solid line ($y_n = x_{n-1} + \alpha \times x_n$, where x_{n-1} is the global loss in $(n-1)$th epoch, y_n is the moving average result of global loss in (n)th epoch, and α is the learning rate). The moving average result of the global loss shows a clearer decreasing trend. In Fig. 11*b*, the training error, validation error, and testing error are plotted in black, blue, and red, respectively. The training errors are evaluated for every five epoches. The network is convergent after 250 epoches.

After the five cross-validation experiments, to further evaluate the classification performance, confusion matrix is used to visualise the discrepancy between the actual class labels and predicted results from the classification. Confusion matrix gives the full picture at the errors made by a classification model. The rows correspond to the known class of the data, that is, the labels in the data. The columns correspond to the predictions made by the model. The value of each of element in the matrix is the number of predictions made with the class corresponding to the column, for example, with the correct value as represented by the row. Thus, the diagonal elements show the number of correct classifications made for each class, and the off-diagonal elements show the errors made. The confusion matrices of our results are shown in Table 1. The accuracy for normal driving, operating shift gear, eating/smoking, and responding to a cell phone are 99.47, 100, 100, and 99.55%, respectively.

From the confusion matrix, pose 2 and pose 3 gain 100% classification accuracy, while around 0.5% of pose 1 and pose 4 are misclassified to pose 3. Fig. 12 shows some misclassification samples. In Fig. 12*a*, the hand position in misclassified pose 1 is very close to mouth. This is the reason that they are misclassified to pose 2 (eating or smoking). In Fig. 12*b*, the hand position is in the middle of head and string wheel. These samples do not belong to any of our defined posture class. The error sample in pose 4 causes the misclassification.

5.4 Evaluation on the new driving posture database

The SEU dataset was recorded under normal lighting conditions in a still vehicle. Low illumination and other realistic driving conditions were not considered. We further evaluated the effectiveness and generalisation performance of our own driving posture database. It consists of two separated datasets, namely, Driving-Posture-atNight and Driving-Posture-inReal. The first one was recorded at night, while the second one was recorded in real driving conditions. Experiments were conducted in the similar procedure as with the SEU dataset. The confusion matrices of evaluation experiment on the Driving-Posture-atNight and the Driving-Posture-inReal are shown in Tables 2 and 3, respectively.

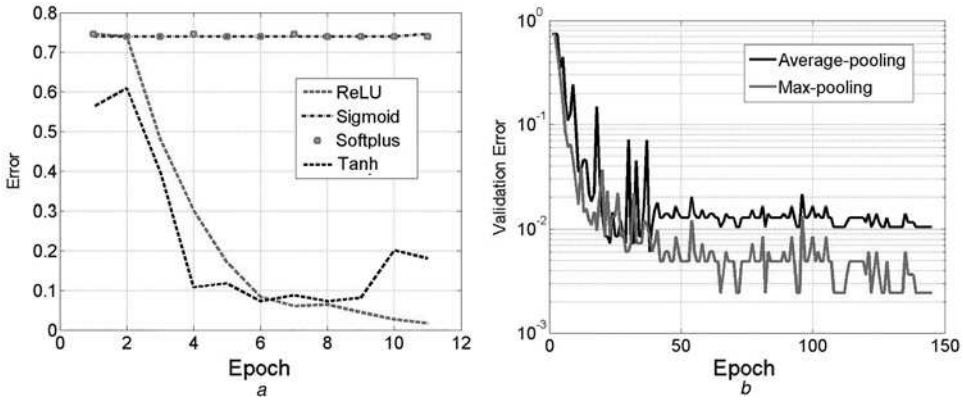

Fig. 10 *Selecting activation function and pooling method*

a Validation error of four activation functions
b Validation error of two pooling method

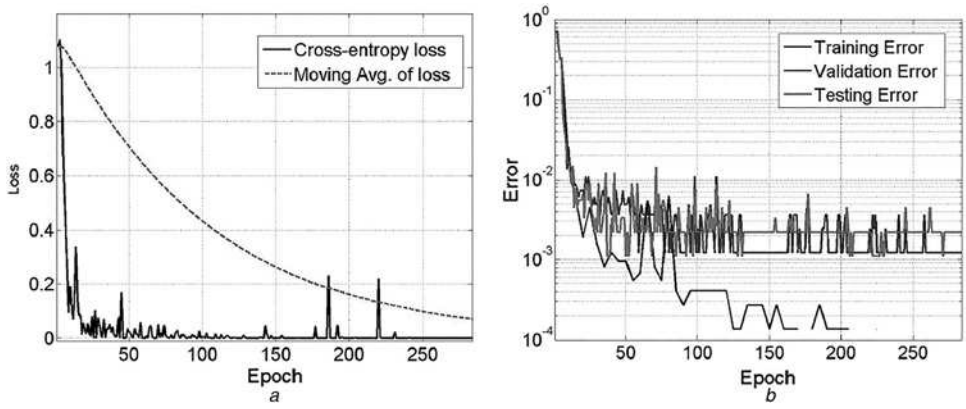

Fig. 11 *Plots of four activation functions*

a Loss value, $\alpha = 0.01$
b Errors

Table 1 Confusion matrix for the cross-validation result

Class	Pose 1	Pose 2	Pose 3	Pose 4
pose 1	99.47	0	0.53	0
pose 2	0	100	0	0
pose 3	0	0	100	0
pose 4	0	0	0.45	99.55

Table 2 Confusion matrix for experiment using Driving-Posture-atNight

Class	Pose 1	Pose 2	Pose 3	Pose 4
pose 1	99.75	0	0.25	0
pose 2	0	100	0	0
pose 3	0	0	99.30	0.70
pose 4	0	0	0.50	99.50

5.5 Comparison with other methods

To provide a comprehensive performance evaluation, we conducted further experiments to compare the proposed CNN approach with several conventional methods using hand-crafted features. Specifically, the compared approaches include: (i) the method proposed in [13], which represents the posture pattern by contourlet transform on skin region; (ii) the method proposed in [56], which extracts feature using mutiwavelet transform method from skin region; (iii) an classifier ensemble method [57]; (iv) Bayesian classifier approach [58]; (v) pyramid histogram of oriented gradients (PHOG) descriptor [59] followed by support vector machine (SVM), and (vi) scale-invariant feature transform (SIFT) descriptor [60] followed by SVM.

For fair comparison, we re-implemented the methods proposed in [13, 56–58] and carried out experiment on other two popular vision descriptor approaches [59, 60]. The approach proposed in [13] and other three approaches [56–58] recognise driving postures using hand and head positions. These methods are sensitive to illumination conditions and different race skin colours, which are located by skin colour segmentation. In other two approaches

Table 3 Confusion matrix for experiment using Driving-Posture-inReal

Class	Pose 1	Pose 2	Pose 3	Pose 4
pose 1	95.77	0	2.70	1.53
pose 2	0.22	99.15	0.63	0
pose 3	1.35	0.82	96.23	1.60
pose 4	0.55	0	1.90	97.55

Table 4 Classification accuracy compared with other six approaches

	Pose 1	Pose 2	Pose 3	Pose 4	Avg.
baseline [13]	97.70	87.55	85.95	89.30	90.63
WT [56]	97.52	92.77	88.99	83.02	89.23
RSE [57]	99.95	91.20	99.20	87.42	94.20
Bayes [58]	94.82	95.20	98.26	92.77	95.11
PHOG + SVM	**99.83**	88.71	89.12	73.20	91.56
SIFT + SVM	99.40	93.52	94.55	91.21	96.12
proposed	99.47	**100**	**100**	**99.55**	**99.78**

Fig. 12 *Misclassification analysis*

a Pose 1 misclassified to pose 3
b Pose 4 misclassified to pose 3

IET Comput. Vis., 2016, Vol. 10, Iss. 2, pp. 103–114

[59, 60], feature are extracted from the whole frame without any pre-processing. The cross-validation experiment procedure was repeated in the same way as we did in Section 5.3. The results are shown in Table 4. It clearly demonstrates that our approach outperforms all of the methods compared.

5.6 Discussion

The proposed CNN approach has been tested on the SEU database and our own database, demonstrating the effectiveness in daytime/night conditions and the generalisation performance in real driving environments. In the comparison experiment, the CNN model offers better performance than other approaches with hand-engineered features. The end-to-end learning model is able to learn the temporal cues and discriminative representation from raw images. However, there exists some limitations with the proposed method. First, training takes a long time even it runs on GPUs such as several days. The algorithm takes high computation resource which makes it difficult to be applied in some conditions with common hardware architecture, for example, off-line embedded system. Second, training a CNN needs a large amount of unconstraint data which is also difficult in some situations.

6 Conclusion

This paper addresses the importance of automatic understanding and characterisation of driver behaviours in the scenario of reducing motor vehicle accidents, and presents a novel system for vision-based driving behaviour recognition. We verified our approach on the SEU driving dataset which includes postures of normal driving, operating the shift gear, eating or smoking, and responding to a cell phone. The proposed approach applied deep CNN, which learns feature from raw image automatically. We have described the details of each layer in our network and evaluated the selection of networks architecture and hyper-parameters. The final results demonstrate better performance than conventional approaches with hand-coded features, achieving an overall accuracy of 99.47% on the SEU dataset, 99.3% on the Driving-Posture-atNight dataset, and 95.77% on the Driving-Posture-inReal dataset.

7 References

1 WHO: 'World report on road traffic injury prevention', 2004. Available at: http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/en/
2 Wu, B.-F., Chen, Y.-H., Yeh, C.-H.: 'Driving behaviour-based event data recorder', *IET Intell. Transp. Syst.*, 2014, **8**, (4), pp. 361–367
3 Jiménez, F., Naranjo, J., Gómez, O.: 'Autonomous collision avoidance system based on accurate knowledge of the vehicle surroundings', *IET Intell. Transp. Syst.*, 2015, **9**, (1), pp. 105–117
4 Tada, M., Noma, H., Utsumi, A., *et al.*: 'Elderly driver retraining using automatic evaluation system of safe driving skill', *IET Intell. Transp. Syst.*, 2014, **8**, (3), pp. 266–272
5 Watta, P., Lakshmanan, S., Hou, Y.: 'Nonparametric approaches for estimating driver pose', *IEEE Trans. Veh. Technol.*, 2007, **56**, (4), pp. 2028–2041
6 Bergasa, L., Nuevo, J., Sotelo, M., *et al.*: 'Real-time system for monitoring driver vigilance', *IEEE Trans. Intell. Transp. Syst.*, 2006, **7**, (1), pp. 63–77
7 Cheng, S.Y., Park, S., Trivedi, M.M.: 'Multi-spectral and multi-perspective video arrays for driver body tracking and activity analysis', *Comput. Vis. Image Underst.*, 2007, **2–3**, (2C3), pp. 245–257
8 Murphy-Chutorian, E., Trivedi, M.: 'Head pose estimation and augmented reality tracking: an integrated system and evaluation for monitoring driver awareness', *IEEE Trans. Intell. Transp. Syst.*, 2010, **11**, (2), pp. 300–311
9 Doshi, A., Trivedi, M.: 'On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes', *IEEE Trans. Intell. Transp. Syst.*, 2009, **10**, (3), pp. 453–462, doi:10.1109/TITS.2009.2026675
10 Teyeb, I., Jemai, O., Zaied, M., *et al.*: 'A drowsy driver detection system based on a new method of head posture estimation', in Corchado, E., Lozano, J., Quintián, H., Yin, H. (Eds.): 'Intelligent data engineering and automated learning C IDEAL 2014', (*LNCS*, 8669) (Springer International Publishing, 2014), pp. 362–369
11 Teyeb, I., Jemai, O., Zaied, M., *et al.*: 'A novel approach for drowsy driver detection using head posture estimation and eyes recognition system based on wavelet network'. The 5th Int. Conf. on Information, Intelligence, Systems and Applications, IISA 2014, 2014, pp. 379–384
12 Ji, Q., Zhu, Z., Lan, P.: 'Real-time nonintrusive monitoring and prediction of driver fatigue', *IEEE Trans. Veh. Technol.*, 2004, **53**, (4), pp. 1052–1068
13 Zhao, C., Zhang, B., He, J., *et al.*: 'Recognition of driving postures by contourlet transform and random forests', *IET Intell. Transp. Syst.*, 2012, **6**, (2), pp. 161–168, doi: 10.1049/iet-its.2011.0116
14 Jemai, O., Teyeb, I., Bouchrika, T., *et al.*: 'A novel approach for drowsy driver detection using eyes recognition system based on wavelet network', *Int. J. Recent Contrib. Eng. Sci. IT (iJES)*, 2013, **1**, (1), pp. 46–52
15 Veeraraghavan, H., Bird, N., Atev, S., *et al.*: 'Classifiers for driver activity monitoring', *Transp. Res. C, Emerg. Technol.*, 2007, **15**, (1), pp. 51–67
16 Cheng, S., Trivedi, M.: 'Vision-based infotainment user determination by hand recognition for driver assistance', *IEEE Trans. Intell. Transp. Syst.*, 2010, **11**, (3), pp. 759–764, doi: 10.1109/TITS.2010.2049354
17 Tran, C., Doshi, A., Trivedi, M.M.: 'Modeling and prediction of driver behavior by foot gesture analysis', *Comput. Vis. Image Underst.*, 2012, **116**, (3), pp. 435–445
18 Hinton, G., Osindero, S., Teh, Y.: 'A fast learning algorithm for deep belief nets', *Neural Comput.*, 2006, **18**, (7), pp. 1527–1554
19 Le, Q., Zou, W., Yeung, S., *et al.*: 'Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2011, 2011, pp. 3361–3368, doi: 10.1109/CVPR.2011.5995496
20 Lecun, Y., Bottou, L., Bengio, Y., *et al.*: 'Gradient-based learning applied to document recognition', *Proc. IEEE*, 1998, **86**, (11), pp. 2278–2324, doi: 10.1109/5.726791
21 Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., *et al.*: 'Convolutional neural networks for speech recognition', *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 2014, **22**, (10), pp. 1533–1545
22 Mao, Q., Dong, M., Huang, Z., *et al.*: 'Learning salient features for speech emotion recognition using convolutional neural networks', *IEEE Trans. Multimed.*, 2014, **16**, (8), pp. 2203–2213, doi: 10.1109/TMM.2014.2360798
23 Hu, B., Lu, Z., Li, H., *et al.*: 'Convolutional neural network architectures for matching natural language sentences', in Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (Eds.): 'Advances in neural information processing systems 27', (Curran Associates Inc., 2014), pp. 2042–2050
24 Krizhevsky, A., Sutskever, I., Hinton, G.E.: 'Imagenet classification with deep convolutional neural networks'. Advances in Neural Information Processing Systems, 2012, pp. 1097–1105
25 Krause, J., Gebru, T., Deng, J., *et al.*: 'Learning features and parts for fine-grained recognition'. Twenty-Second Int. Conf. on Pattern Recognition (ICPR), 2014, 2014, pp. 26–33, doi: 10.1109/ICPR.2014.15
26 Simonyan, K., Zisserman, A.: 'Two-stream convolutional networks for action recognition in videos', in Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (Eds.): 'Advances in neural information processing systems 27' (Curran Associates Inc., 2014), pp. 568–576
27 Zhang, N., Paluri, M., Ranzato, M., *et al.*: 'Panda: pose aligned networks for deep attribute modeling'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, 2014, pp. 1637–1644
28 Girshick, R., Donahue, J., Darrell, T., *et al.*: 'Rich feature hierarchies for accurate object detection and semantic segmentation'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, 2014, pp. 580–587
29 Farabet, C., Couprie, C., Najman, L., *et al.*: 'Learning hierarchical features for scene labeling', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013, **35**, (8), pp. 1915–1929, doi: 10.1109/TPAMI.2012.231
30 Weinzaepfel, P., Revaud, J., Harchaoui, Z., *et al.*: 'Deepflow: large displacement optical flow with deep matching'. IEEE Int. Conf. on Computer Vision (ICCV), 2013, 2013, pp. 1385–1392, doi: 10.1109/ICCV.2013.175
31 Yi, D., Lei, Z., Liao, S., *et al.*: 'Deep metric learning for person re-identification'. Twenty-Second Int. Conf. on Pattern Recognition (ICPR), 2014, 2014, pp. 34–39
32 Taigman, Y., Yang, M., Ranzato, M., *et al.*: 'Deepface: closing the gap to human-level performance in face verification'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, 2014, pp. 1701–1708
33 Sun, Y., Wang, X., Tang, X.: 'Deep learning face representation from predicting 10,000 classes'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, 2014, pp. 1891–1898, doi: 10.1109/CVPR.2014.244
34 Ngiam, J., Chen, Z., Bhaskar, S.A., *et al.*: 'Sparse filtering', in Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.): 'Advances in neural information processing systems 24' (Curran Associates Inc., 2011), pp. 1125–1133
35 Glorot, X., Bordes, A., Bengio, Y.: 'Deep sparse rectifier neural networks'. Journal of Machine Learning Research 15 (Proc. 14th Int. Conf. on Artificial Intelligence and Statistics, AISTATS 2011), 2011, pp. 315–323
36 Jin, J., Fu, K., Zhang, C.: 'Traffic sign recognition with hinge loss trained convolutional neural networks', *IEEE Trans. Intell. Transp. Syst.*, 2014, **15**, (5), pp. 1991–2000, doi: 10.1109/TITS.2014.2308281
37 Hubel, D.H., Wiesel, T.N.: 'Receptive fields of single neurones in the cat's striate cortex', *J. Physiol.*, 1959, **148**, pp. 574–591
38 Bengio, Y., Simard, P., Frasconi, P.: 'Learning long-term dependencies with gradient descent is difficult', *IEEE Trans. Neutral Netw.* 1994, **5**, (2), pp. 157–166
39 Dugas, C., Bengio, Y., Bélisle, F., *et al.*: 'Incorporating second-order functional knowledge for better option pricing', in Leen, T., Dietterich, T., Tresp, V. (Eds.): 'Advances in neural information processing systems 13' (MIT Press, 2001), pp. 472–478
40 Boureau, Y.-L., Ponce, J., Lecun, Y.: 'A theoretical analysis of feature pooling in visual recognition'. Twenty-Seventh Int. Conf. on Machine Learning, Haifa, Israel, 2010
41 Zeiler, M.D., Fergus, R.: 'Stochastic pooling for regularization of deep convolutional neural networks', Available at: http://arxiv.org/abs/1301.3557 abs/1301.3557
42 Jarrett, K., Kavukcuoglu, K., Ranzato, M., *et al.*: 'What is the best multi-stage architecture for object recognition?'. IEEE 12th Int. Conf. on Computer Vision, 2009, pp. 2146–2153, doi: 10.1109/ICCV.2009.5459469

43 Dong, Z., Pei, M., He, Y., *et al.*: 'Vehicle type classification using unsupervised convolutional neural network'. Twenty-Second Int. Conf. on Pattern Recognition (ICPR), 2014, pp. 172–177, doi: 10.1109/ICPR.2014.39
44 Simoncelli, E.: 'Statistical models for images: compression, restoration and synthesis'. Conf. Record of the Thirty-First Asilomar Conf. on Signals, Systems Computers 1997, 1997, vol. 1, pp. 673–678
45 Bethge, M.: 'Factorial coding of natural images: how effective are linear model in removing higher-order dependencies?', *J. Opt. Soc. Am. A*, 2006, **23**, (6), pp. 1253–1268
46 Pinto, N., Cox, D.D., DiCarlo, J.J.: 'Why is real-world visual object recognition hard?', *PLOS Comput. Biol.*, 2008, **4**, (1), p. 27
47 Lyu, S., Simoncelli, E.: 'Nonlinear image representation using divisive normalization'. IEEE Conf. on Computer Vision and Pattern Recognition, 2008. CVPR 2008, 2008, pp. 1–8, doi:10.1109/CVPR.2008.4587821
48 Rumelhart, D.E., Hinton, G.E., Williams, R.J.: 'Learning representations by back-propagating errors', *Nature*, 1986, **323**, (6088), p. 533
49 Murphy, K.P.: 'Machine learning: a probabilistic perspective, adaptive computation and machine learning' (MIT Press, Cambridge, Mass, 2012)
50 Online, http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
51 Erhan, D., Bengio, Y., Courville, A., *et al.*: 'Why does unsupervised pre-training help deep learning?', *J. Mach. Learn. Res.*, 2010, **11**, pp. 625–660
52 Yosinski, J., Clune, J., Bengio, Y., *et al.*: 'How transferable are features in deep neural networks?', in Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (Eds.): 'Advances in neural information processing systems 27' (Curran Associates Inc., 2014), pp. 3320–3328
53 Vincent, P., Larochelle, H., Bengio, Y., *et al.*: 'Extracting and composing robust features with denoising autoencoders'. Proc. Twenty-Fifth Int. Conf. on Machine Learning (ICML 2008), Helsinki, Finland, 5–9 June 2008, pp. 1096–1103
54 Olshausen, B.A., Fieldt, D.J.: 'Sparse coding with an overcomplete basis set: a strategy employed by v1 ?
55 Lee, H., Ekanadham, C., Ng, A.Y.: 'Sparse deep belief net model for visual area v2', in Platt, J., Koller, D., Singer, Y., Roweis, S. (Eds.): 'Advances in neural information processing systems 20' (Curran Associates Inc., 2008), pp. 873–880
56 Zhao, C., Gao, Y., He, J., *et al.*: 'Recognition of driving postures by multiwavelet transform and multilayer perceptron classifier', *Eng. Appl. Artif. Intell.*, 2012, **25**, (8), pp. 1677–1686
57 Zhao, C., Zhang, B., Zhang, X., *et al.*: 'Recognition of driving postures by combined features and random subspace ensemble of multilayer perceptron classifiers', *Neural Comput. Appl.*, 2013, **22**, (1), pp. 175–184
58 Zhao, C., Zhang, B., He, J.: 'Vision-based classification of driving postures by efficient feature extraction and bayesian approach', *J. Intell. Robot. Syst.*, 2013, **72**, (3–4), pp. 483–495, doi: 10.1007/s10846-012-9797-z
59 Bosch, A., Zisserman, A., Munoz, X.: 'Representing shape with a spatial pyramid kernel'. Proc. 6th ACM Int. Conf. on Image and Video Retrieval, CIVR '07, ACM, New York, NY, USA, 2007, pp. 401–408
60 Lowe, D.: 'Distinctive image features from scale-invariant keypoints', *Int. J. Comput. Vis.*, 2004, **60**, (2), pp. 91–110, doi: 10.1023/B:VISI.0000029664.99615.94