Applied Soft Computing Journal 74 (2019) 1–9

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Driver identification based on hidden feature extraction by using adaptive nonnegativity-constrained autoencoder

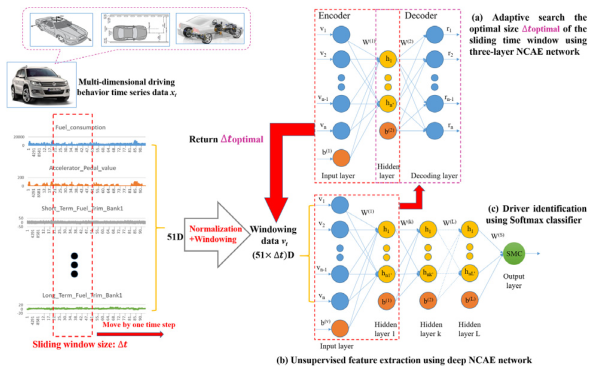
Jie Chen [a,b,*], ZhongCheng Wu [a,c], Jun Zhang [a]

[a] *Hefei Institute of Physical Science, Chinese Academy of Sciences, Hefei, China*
[b] *Graduate School of Computer Applied Technology, University of Science and Technology of China, Hefei, China*
[c] *University of Science and Technology of China, Hefei, China*

HIGHLIGHTS

- We proposed a three-layer autoencoder to adaptively search time window size.
- We constructed a deep autoencoder to automatically extract the hidden features.
- The proposed method is superior to the existing state-of-the-art methods.

GRAPHICAL ABSTRACT


Multi-dimensional driving behavior time series data x_i
Fuel_consumption
Accelerator_Pedal_value
Short_Term_Fuel_Trim_Bank1
Long_Term_Fuel_Trim_Bank1
Move by one time step
Sliding window size: Δt
51D
Normalization +Windowing
Windowing data v_i (51× Δt)D
Return Δtoptimal
Encoder
Decoder
Input layer
Hidden layer
Decoding layer
(a) Adaptive search the optimal size Δtoptimal of the sliding time window using three-layer NCAE network
(c) Driver identification using Softmax classifier
SMC
Output layer
Hidden layer 1
Hidden layer k
Hidden layer L
(b) Unsupervised feature extraction using deep NCAE network

ARTICLE INFO

Article history:
Received 21 May 2018
Received in revised form 5 September 2018
Accepted 23 September 2018
Available online xxxx

Keywords:
Nonnegativity-constrained autoencoder
Driver identification
Adaptive search
Deep learning
Feature extraction

ABSTRACT

In this paper, we propose a new driver identification method using deep learning. Existing driver identification methods have the disadvantages that the size of the sliding time window is too large and the feature extraction is relatively subjective, which leads to low identification accuracy and long prediction time. We first propose using an unsupervised three-layer nonnegativity-constrained autoencoder to adaptive search the optimal size of the sliding window, then construct a deep nonnegativity-constrained autoencoder network to automatically extract hidden features of driving behavior to further complete driver identification. The results from the public driving behavior dataset indicate that relative to conventional sparse autoencoder, dropout-autoencoder, random tree, and random forest algorithms, our method can effectively search the optimal size of the sliding time window, and the window size is shortened from the traditional 60s to 30s, which can better preserve the intrinsic information of the data while greatly reducing the data volume. Furthermore, our method can extract more distinctive hidden features that aid the classifier to map out the separating boundaries among the classes more easily. Finally, our method can significantly shorten the prediction time and improve the timeliness under the premise of improving the driver identification performance and reducing the model overfitting.

1. Introduction

Driver identification, a key topic in the field of driving behavior analysis, has important research value in many fields [1,2]. The most widely used area for driver identification is car safety [3–8], that is, vehicle management companies use different technical

[*] Corresponding author at: Hefei Institute of Physical Science, Chinese Academy of Sciences, Hefei, China.
E-mail addresses: cj2016@mail.ustc.edu.cn (J. Chen), zcwu@iim.ac.cn (Z. Wu), zhang_jun@hmfl.ac.cn (J. Zhang).

https://doi.org/10.1016/j.asoc.2018.09.030
1568-4946/© 2018 Elsevier B.V. All rights reserved.

methods to monitor and verify whether the driver is the owner of the car, to prevent theft. Other actors that can benefits from the driver identification are the insurance companies: new insurance paradigms, as the "Usage-based insurance (UBI)", are emerging [9,10]. UBI, which is based on the original auto insurance, integrates driving behavior information to accurately price auto insurance premiums for drivers of different driving styles, and encourages safe driving. However, successful driver identification is the first step to complete the above driving behavior analysis and risk assessment [11,12].

Current research related to driver identification mainly includes the following aspects:

(1) Early driver identification studies mostly focused on biometrics such as passwords, fingerprinting, iris recognition, and face recognition [8]. These methods have difficulty in large-scale application to driver identification because of low recognition accuracy and privacy security problems caused by biological information.
(2) With the advancement of smart interconnection, vehicle-mounted On Board Diagnostics (OBD) is widely used, and it is possible to access massive driving behavior data [9,13,14]. Researchers have begun to focus on driver identification research based on massive driving behavior data [7,8]. There are two main aspects: the first is driver identification based statistical characteristics of historical driving behavior data [12,15,16]. Researcher uses specific braking, acceleration and speeding frequencies as features to distinguish different drivers [12]. The other is driver identification based on time series data [7,17–20]. The basic idea is using the original driving time series data as a feature candidate set, and combined with principal component analysis (PCA) and other methods for feature sorting to select the optimal feature set, and then use classifiers to complete the driver identification. There are also researchers who combine time series data and statistical features (including mean, median and standard deviation) for driver identification [8].

In this context, the driver identification methods based on driving behavior data in (2) will gradually become mainstream, but the current research methods still have the following problems:

(a) The problem of selecting the size of the sliding time window. Current research adopted 60 s as the minimum size of the sliding time window for the original time series data, and result in excessive data volume. How to ensure the classification accuracy while adaptive searching a more appropriate minimum size of the sliding time window still needs to be resolved;
(b) Most of the feature selection methods used in the research, such as statistics, PCA, Ranker search methods, etc., have a strong subjectivity and are not representative;
(c) In the current researches, the raw multi-dimensional time series data and statistical features are combined into feature sets, and lead to large feature dimension, excessive training time, low prediction for the model, and poor real-time performance.

In view of the deficiencies of the existing research methods, firstly, we propose a new adaptive method to search the optimal size of the sliding time window, then we construct an nonnegativity constrained autoencoder (NCAE) [21–25] for unsupervised learning to automatically extract more representative and distinct hidden features of driving behavior, finally, we improve the driver identification performance with softmax classifier. The main contributions of this work lie in the following aspects:

(i) First, to solve the problem of uncertain time window size, a three-layer NCAE network is constructed to adaptively search the optimal size of the sliding time window. The reconstruction error of the raw driving behavior data is used as the goal orientation to perform unsupervised minimization optimization to search the optimal size of the sliding time window;
(ii) Further, we build a Deep NCAE (DNCAE) network to automatically extract the 10-dimensional hidden features of driving behavior to eliminate the subjectivity of the manual feature selection and reduce the feature dimension;
(iii) Finally, the results of cross-validation experiments indicate that the proposed DNCAE model can significantly improve the driver identification performance, lower the overfitting, and at the same time shorten the prediction time and improve the timeliness.

2. Methods

2.1. Adaptive search the optimal size of sliding time window using three-layer NCAE

The general schematic of adaptive search the optimal size of the sliding time window using NCAE is shown in Fig. 1. At the first, as seen in the left part of Fig. 1, we define the dataset of driving behavior as $\mathrm{X} \in \mathbb{R}^{D_X \times N_X}$. Each time step t of data is defined as

$$\mathrm{x}_t = \left(x_{t,1}, x_{t,2}, \ldots, x_{t,D_X}\right)^T \in \mathbb{R}^{D_X} \tag{1}$$

where D_X is the dimensionality of x_t, and N_X is the amount of dataset X, i.e., the total number of time steps of X. We consider the driving information for each dimension of a unit is different, so we use the maximum and minimum values of each dimension to normalize in each dimension independently to (0,1). The normalized data $\mathrm{n}_t \in \mathbb{R}^{D_X}$ for the t-step are

$$\mathrm{n}_t = \left(n_{t,1}, n_{t,2}, \ldots, n_{t,D_X}\right)^T \in \mathbb{R}^{D_X} \tag{2}$$

$$n_{t,d} = \frac{x_{t,d} - x_{d\ min}}{x_{d\ max} - x_{d\ min}} \tag{3}$$

where $x_{d\ max}$ and $x_{d\ min}$ are the minimum and maximum values of the d-th dimension for all data x in dataset X. To extract contextual features and mitigate the ill effects of noises, we use windowing process to set a time window Δt. The windowed data v_t have $D_V = \Delta t \times D_X$ dimensions, which are generated by

$$\mathrm{v}_t = \left(\mathrm{n}_{t-\Delta t+1}, \mathrm{n}_{t-\Delta t+2}, \ldots, \mathrm{n}_t\right)^T \in \mathbb{R}^{D_V} (t \geq \Delta t) \tag{4}$$

when v_t moves at the time axis by one time step, we can obtain a windowing dataset $\mathrm{V} \in \mathbb{R}^{D_V \times N_V}$, which has $N_V = N_X - \Delta t + 1$ time steps.

To avoid excessive data volume and ensure prediction accuracy, it is very important to choose an optimal size of the sliding time window. In this paper, we construct an NCAE neural network and conduct unsupervised pre-training to search the optimal Δt value, as shown in the right part of Fig. 1. An autoencoder (AE) is an unsupervised feature learning framework that tries to reconstruct its input vector at the output through unsupervised learning [26–28]. The network tries to learn a function

$$\mathrm{r}_t = f_{\mathrm{w,b}}(\mathrm{v}_t) \approx \mathrm{v}_t \tag{5}$$

where v_t is the input vector, r_t is the reconstructed vector, and $f(\cdot)$ is activation function. In addition, $\mathrm{W} = \{\mathrm{W}^{(1)}, \mathrm{W}^{(2)}\}$ and $\mathrm{b} = \{\mathrm{b}^{(1)}, \mathrm{b}^{(2)}\}$ represent the weights and biases of the encoding and decoding layers, respectively.

To optimize the parameters of the model in (5), the average reconstruction error is used as the cost function

$$J_{AE}(\mathrm{W}, \mathrm{b}) = \frac{1}{2N_V} \sum_{t=1}^{N_V} \|r_t - v_t\|^2 \tag{6}$$

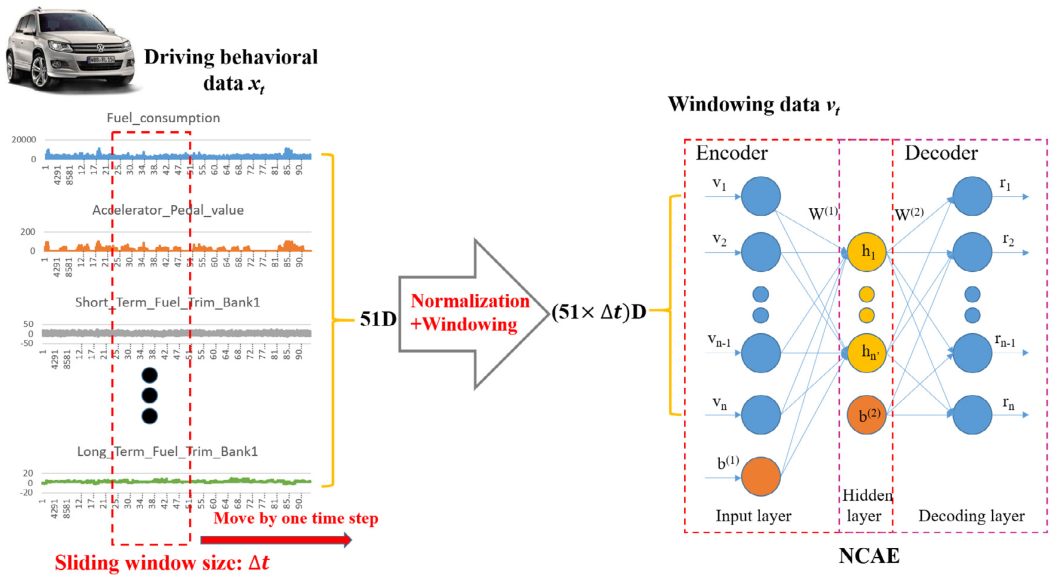


Fig. 1. Schematic diagram of adaptively search the optimal size of the sliding time window using a three-layer NCAE.

where N_V is the number of training samples.

To prevent overfitting and to extract non-negative latent features, an $L2$ non-negative weight decay term is introduced [29]. In addition, we use the Kullback–Leibler (KL) divergence [30,31] to obtain more sparse and prominent features. Therefore, we ultimately obtain the following objective function for the NCAE [32]

$$J_{NCAE}(\mathrm{W},\mathrm{b}) = J_{AE}(\mathrm{W},\mathrm{b}) + J_N({\omega_{ij}}^{(l)}) + \beta J_{KL}(p \parallel \hat{p}) \tag{7}$$

where

$$J_N({\omega_{ij}}^{(l)}) = \frac{\alpha}{2}\sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l-1}} \begin{cases} ({\omega_{ij}}^{(l)})^2 & {\omega_{ij}}^{(l)} < 0 \\ 0 & {\omega_{ij}}^{(l)} \geq 0 \end{cases} \tag{8}$$

$$J_{KL}(p \parallel \hat{p}) = \sum_{j=1}^{n'} p\log\frac{p}{p_j} + (1-p)\log\frac{1-p}{p_j} \tag{9}$$

The first term on the right side of (7) is the reconstruction error. The second term represents the weight penalty term a that can control the strength of the penalty term ${\omega_{ij}}^{l}$ that expresses the connection between the jth unit in layer $l-1$ and the ith unit in layer l. The third term is the sparse item β used to adjust the size of the term. n' is the number of hidden layer nodes and p is the sparsity target chosen to be a small positive number near 0. The average activation of this hidden unit is

$$p_j = \frac{1}{N_V}\sum_{t=1}^{N_V} h_j(\mathrm{v}_t) \tag{10}$$

To minimize this constrained objective function (7), we use the gradient descent algorithm to update the weights and biases. The gradient update is shown below:

$${\omega_{ij}}^{(l)} = {\omega_{ij}}^{(l)} - \lambda\frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_{NCAE}(W,b) \tag{11}$$

$${b_i}^{(l)} = {b_i}^{(l)} - \lambda\frac{\partial}{\partial{b_i}^{(l)}}J_{NCAE}(W,b) \tag{12}$$

where $\lambda > 0$ is the learning rate. The gradient of J_{NCAE} (W, b) is computed as follows

$$\frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_{NCAE}(\mathrm{W},\mathrm{b}) = \frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_{AE}(\mathrm{W},\mathrm{b}) + \frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_N({\omega_{ij}}^{(l)}) + \beta\frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_{KL}(p \parallel \hat{p}) \tag{13}$$

The gradient of the $L2$-weight decay term is

$$\frac{\partial}{\partial{\omega_{ij}}^{(l)}}J_N({\omega_{ij}}^{(l)}) = \begin{cases} \alpha{\omega_{ij}}^{(l)} & {\omega_{ij}}^{(l)} < 0 \\ 0 & {\omega_{ij}}^{(l)} \geq 0 \end{cases} \tag{14}$$

Based on the NCAE network, we propose an adaptive search approach for the optimal size of sliding windows as shown in **Algorithm 1**. Where data (Δt) is the input matrix of the network, that is v_t , and Δt is the output value of the **Algorithm 1**, which is the window size that we need to search. Δt_0 is the initial time window size. W (Δt_{init}) and b (Δt_{init}) are the network initial weights and biases respectively for the time window size of Δt. It is worth mentioning that the dimension of the input matrix v_t changes when the Δt value is updated during network training, at the same time, we also need to update the dimensions of W (Δt), that is, we update the number of hidden layer nodes (n') to be half the dimension of the input matrix v_t in each loop (It should be noted that we set the number of hidden nodes to half of the number of input nodes based on previous research experience and the deep learning references [33]), and perform the same initialization for W (Δt). In addition, *pretrain_iter* is the number of initial iterations for network training, and *scan_iter* is the number of iterations for network update optimization.

2.2. Feature extraction and driver identification using DNCAE network

Based on Section 2.1, the optimal time window size $\Delta t_{\text{optimal}}$ was determined through a three-layer NCAE. Then, we constructed a multi-layer NCAE network to extract the hidden features of driving behavior and further realize the driver identification.

In the construction of the deep network (DN), we first stack the multi-layer NCAE into a deep architecture, Second, we use a greedy layer-wise training method to extract the hidden features of each layer [31]. The complete feature extraction process can be expressed as: $v_t \rightarrow {h_t}^{(1)} \rightarrow {h_t}^{(2)} \rightarrow \cdots \rightarrow {h_t}^{(final-1)} \rightarrow {h_t}^{(final)}$. The ${h_t}^{(final)}$ is the extracted hidden feature of v_t, which is used as the input into the softmax layer. The structure of the DN is shown in

Algorithm 1 **Adaptive search the optimal size of windows (ASOSW)**

```
    // Input: The windowed data ($\Delta t$), the constructed NCAE model
    // Output: The optimal time window size $\Delta t$
1:  Function $\Delta t$ = ASOSW (data($\Delta t$), NCAE)
2:      initialize: $\Delta t_0$, data ($\Delta t_0$), pretrain_iter, scan_iter
3:      initialize: W($\Delta t_{0\,init}$), b($\Delta t_{0\,init}$)
4:      W($\Delta t_0$),b($\Delta t_0$) = Train_NCAE (data($\Delta t_0$), W($\Delta t_{0\,init}$), b($\Delta t_{0\,init}$), pretrain_iter);
5:      $J_{AE}$ ($\Delta t_0$) = Compute_Loss (data($\Delta t_0$), W($\Delta t_0$), b($\Delta t_0$)) as in (7);
6:    for $\Delta t$ = 2 to 240 do{
7:       W($\Delta t$),b($\Delta t$)=Train_NCAE(data($\Delta t$), W($\Delta t_{init}$), b($\Delta t_{init}$), scan_iter) ;
8:       $J_{AE}$ ($\Delta t$)= Compute_Loss (data ($\Delta t$), W($\Delta t$), b($\Delta t$));
9:      if $J_{AE}$ ($\Delta t$) $\leq$ $J_{AE}$ ($\Delta t_0$) then
10:          $J_{AE}$ ($\Delta t_0$) = $J_{AE}$ ($\Delta t$);
11:      else
12:          $\Delta t = \Delta t - 1$
13:            return $\Delta t$
14:      end if
15:    end for}
16: end function
```

Fig. 2, we define the objective function of the NCAE constrained softmax classifier as

$$J_{NC-soft\max}(\mathrm{W}) = J_{soft\max}(\mathrm{W}) + J_N({\omega_{ij}}^{(L)}) \tag{15}$$

where

$$\begin{aligned} J_{soft\max}(\mathrm{W}) &= -\frac{1}{N_V}\left[\sum_{t=1}^{N_V}\sum_{p=1}^{k} \mathrm{I}(y_t = p)\mathrm{logp}(k|y_t)\right] \\ &= -\frac{1}{N_V}\left[\sum_{t=1}^{N_V}\sum_{p=1}^{k} \mathrm{I}(y_t = p)\log\frac{e^{w_p^T h_t^{(final)}}}{\sum_{l=1}^{k} e^{w_l^T h_t^{(final)}}}\right] \end{aligned} \tag{16}$$

$$J_N({\omega_{ij}}^{(L)}) = \frac{\alpha}{2}\sum_{i=1}^{k}\sum_{j=1}^{s_L}\begin{cases} ({\omega_{ij}}^{(L)})^2 & {\omega_{ij}}^{(L)} < 0 \\ 0 & {\omega_{ij}}^{(L)} \geq 0 \end{cases} \tag{17}$$

Here, N_V is the number of samples, k is the number of classes, which is the number of driver class in this paper, w_p is the pth column of weights matrix referring to the input weights of the pth softmax node, y_t represents the label of the tth sample, and I(s) is the indication function, that is, when s is true, I(s) = 1, otherwise 0. p (k |y_t) is the output probability calculated by SMC. s_L is the number of hidden nodes of the final AE.

After completing the greedywise training on the stacked NCAE and the softmax layers, we fine-tune the pretrained DN using the backpropagation algorithm to correct the network parameters to achieve better classification accuracy [31] (It should be noted that the backpropagation is the core of the stochastic gradient descent algorithm. We use the backpropagation algorithm to calculate the gradient according to the chain rule, and then substitute it into the gradient update formula to update the weights and biases of the network. Using the backpropagation algorithm, the gradient values of each node can be obtained faster, and the training efficiency of the model can be improved. In addition, fine-tuning is to better use the verification dataset to optimize the model parameters, so that the model has better generalization ability). The objective function for fine-turning the DN is given by

$$J_{\mathrm{DN}}(\mathrm{W}, \mathrm{b}) = J_{soft\max}(\mathrm{W_{DN}}, \mathrm{b_{DN}}) + J_N({\omega_{ij}}^{(l)}) \tag{18}$$

where $\mathrm{W_{DN}}$ contains the input weights of the NCAE and softmax layers and $\mathrm{b_{DN}}$ is the bias of the NCAE layers.

3. Experimental results

In this section, we provide a set of experiments to verify the performance of the proposed method. All experiments were performed on an Intel(R) Core(TM) i5-7500 CPU @ 3.40 GHz with 8 GB of RAM running the 64-bit Windows 10 Enterprise edition. The software implementation used was MATLAB R2017b (The MathWorks Inc, Natick, Massachusetts, United States) , and the L-BFGS in minFunc [34] is used to minimize the objective functions (7), (15), and (18).

3.1. Datasets

The following driving dataset is used to validate the experiments, and it is freely available for research purpose [7,8]. The data are processed from in-vehicle CAN data. This research uses the On Board Diagnostics 2 (OBD-II) and CarbigsP as OBD-II scanner for data collection. Ten different drivers participated to the experiment by driving, with the same car, 4 different round-trip paths in Seoul (i.e., between Korea University and SANGAM World Cup Stadium) for about 23 h of total driving time. The driving paths are of three types: city way, motor way and parking space with a total length of about 46 km. The ten drivers, labeled from "A" to "J", completed two round trips for reliable classification, while data are collected from totally different road conditions. This dataset sampling frequency is 1 s and a total of 51 driving behavior features are collected through the OBD-II.

Since in the above-mentioned public dataset, each driver repeatedly travels a plurality of times according to a specified route to collect driving behavior time series data, that is, each driver contains data of multiple trips, and the data will be recorded again from 0 at the beginning of each trip. Therefore, the time interval in the same trip is fixed (sampling frequency is 1 s) and continuous, and the time between different trips is not continuous. Because

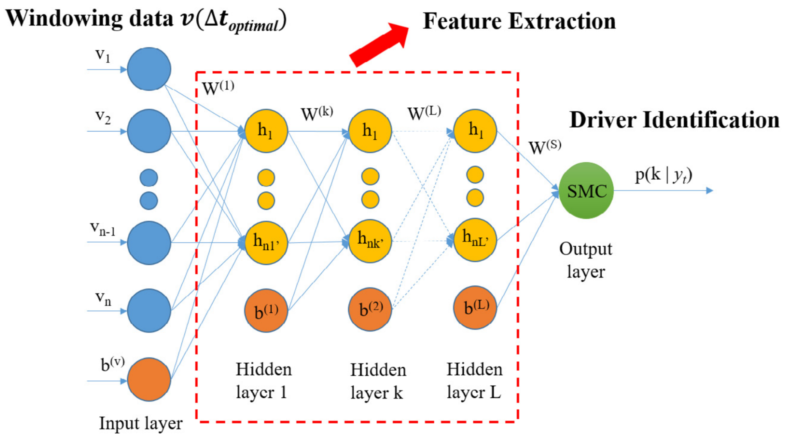


Fig. 2. Schematic diagram of a DNCAE of L + 1 layers constructed using stacked NCAE and Softmax Classifier (SMC).

we are concerned with the processing of continuous time series data, we use the single trip of each driver as the basic unit, and then windowing each trip to obtain the original windowed sample dataset.

We randomly divide the windowed dataset into 10 random parts. 8 parts are selected as the training sample set, and the remaining 1 part is used as the validation sample set, and the other 1 part is used as the test sample set. To evaluate the performance of the proposed model described in this article, training-validation-test strategy is used. Training of the model is performed based on the training set, and the validation set is adopted for tuning the hyperparameters. The overall performance of model is evaluated from the test dataset.

3.2. Experiment I: The impact of hyperparameters

As described in Section 2, in the proposed formulation (i.e. Eqs. (7), (8), (17)), the nonnegativity constraint penalty (a), the sparsity penalty (β), the sparsity parameter (p), and the convergence tolerance respectively control different penalty items, and the settings of these hyperparameters play an important role in the performance of the model.

Therefore, we transform the hyperparameters of different magnitudes to observe their effects on the performance of the model, and the reconstruction error J_{AE} (Eq. (6)) is calculated as an evaluation metric. We used the nonnegativity constraint penalty (a) as an example to visualize the experimental results, so we have performed experiments by varying a in the range of 0 to 1 with varying step sizes, i.e. $a = \{(0.00001{:}0.00001{:}0.0001), (0.0001{:}0.0001{:}0.001), (0.001{:}0.001{:}0.01), (0.01{:}0.01{:}0.1), (0.1{:}0.1{:}1)\}$. As shown in Fig. 3, on the training dataset, we can find that the hyperparameters a is set in the range of 0.001–0.01, which can achieve better reconstruction performance, and the lowest reconstruction error is 0.4264, which is obtained with $a = 0.003$. Similarly, for several other hyperparameters, we use the same test method and select the values corresponding to their respective minimum reconstruction errors as hyperparameters set-values.

Thus, based on the above experimental results, we set the hyperparameters for the subsequent verification experiments. The Nonnegativity constraint penalty (a), the sparsity penalty (β), the sparsity parameter (p), and the convergence tolerance was set to 3e−3, 3, 0.05, and 1e−9, respectively.

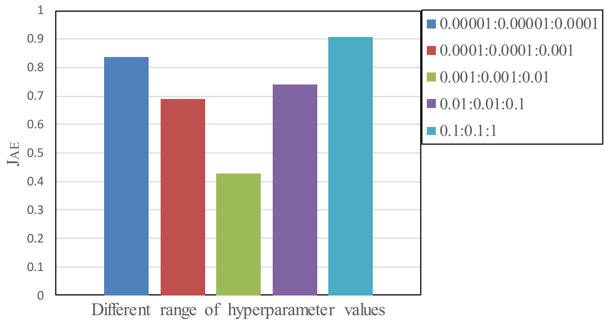


Fig. 3. Effect of a on the training dataset. For each of the five intervals, lowest reconstruction error is reported.

3.3. Experiment II: Adaptive search the optimal size of sliding time window

In this section, we construct a three-layer NCAE network for unsupervised learning and determine the optimal time window size by using **Algorithm 1**.

First, we set the relevant parameters involved in **Algorithm 1**. The initial time window size Δt_0 was set to 1, *pretrain_iter* and *scan_iter* was set to 50, respectively. In experiment, Δt was updated from 2 to 240 and the unit is seconds (s). Therefore, according to the time window size Δt, the size of the windowing data input to the network is expressed as $51\times\Delta t$, then we set the number of hidden layer nodes (n') to be $51\times\Delta t/2$ (if the value is not an integer, we round it), i.e., when the initial Δt_0 is 1 s, the vector dimension of the input v_t is $51 \times 1 = 51$D, and the n' is $51/2 \approx 26$. Thus, we build the three-layer NCAE structure for the $(51 \times\Delta t) - (51 \times\Delta t/2) - (51 \times\Delta t)$ configuration.

We performed 10-fold cross-validation experiments to search the optimal size of the sliding time window on the training sample set. The optimal size $\Delta t_{\text{optimal}}$ of the sliding window obtained by 10-fold cross-validation using **Algorithm 1** is **30.40 s** ± **1.08 s** (that is mean ± standard deviation), therefore, to facilitate the calculation, we determine **30 s** as the optimal size of the sliding time window.

Furthermore, to more intuitively display the optimization process of **Algorithm 1**, we set the proportion of different number of hidden nodes, and the reconstruction error with different window sizes is shown in Fig. 4. It should be noted that the ratio = the number of hidden nodes/the number of input nodes in Fig. 4.

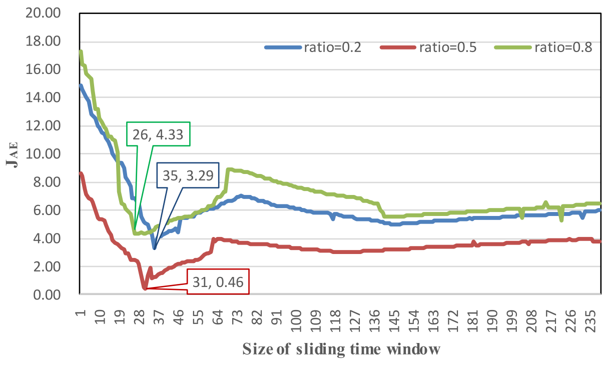


Fig. 4. Reconstruction error using Algorithm 1 with the various size of sliding time window on the training dataset.

First, we can compare the subtle changes of the three curves and find that the reconstruction error when ratio = 0.2 is generally lower than the reconstruction error when ratio = 0.8. The reconstruction performance of ratio = 0.5 used in this paper is better. This result indicates that in the process of setting the number of hidden nodes in the deep network, it is necessary to select a suitable ratio, and selecting too few or too many nodes may not achieve better results. Combined with the above analysis results, we verify the validity of the hidden node ratio set to 0.5 in this paper.

In addition, we can see from Fig. 4 that in the process of optimization, the curves of three different colors all show similar trends, that is, the reconstruction error J_{AE} first gets the global minimum near 30 s, then J_{AE} will have a period of increase and volatility, in the end, it tends to be stable. This result verifies that the optimal window size of 30 s obtained by **Algorithm 1** above is effective. Also shows that the window size needs to be selected within a suitable range. If the window size is too large or too small, the inherent information of the original driving behavior data will be lost, and the information in the window cannot represent the original data well.

Therefore, the 30 s of window size determined based on the proposed unsupervised search algorithm is more appropriate, it can effectively reduce the reconstruction error, and better preserve the original data information. In addition, compared with the conventional window size of 60 s [7,8], the data dimensions effectively reduced, and the timeliness is improved.

3.4. Experiment III: Feature extraction and driver identification

In this section, a series of experiments on the driving behavior dataset are used to evaluate that the proposed method can effectively extract the implied features of driving behavior and the performance of driver identification. We first construct a DNCAE network to extract the hidden features of driving behavior, and then to achieve driver classification with the softmax classifier.

3.5. Unsupervised feature learning: Feature extraction

According to Section 3.2, **30 s** as the optimal size of the sliding time window, and the window move length is 1 s, then the vector dimension input to NCAE is $51 \times 30 = 1530$D. Thus, in the first experiment, based on previous research, selecting the best performing network structure for the feature extraction [32,35,36], we build the DNCAE structure for the 1530-765-380-190-85-40-10 configuration. Firstly, we used the training sample set to conduct unsupervised pretraining on six NCAEs to learn the weights. Then, we stacked six NCAEs to obtain the hidden activities of the last NCAE (that is, hidden features) using the test sample set.

To evaluate the ability of the NCAE network to disentangle the hidden features of driving behavior from the dataset, we use t-distributed stochastic neighbor embedding (t-SNE) to project the high-level representation of hidden features of driving behavior to 2D space [37].

The distribution of features of raw windowing data, NCAE, reference methods (PCA) [7,8] are respectively visualized in Fig. 5(a), (b), and (c). There are two aspects that need to be specially noted: (1) We use t-SNE only to project the features extracted by different methods into the 2D space in the same way, and then we can more intuitively compare the effects of different feature extraction methods. t-SNE is just an efficient visualization tool, and it is not related to the feature extraction and identification methods proposed in this paper. and it is not related to our proposed method and final experimental results. (2) the horizontal and vertical coordinates in Fig. 5 represent the two-dimensional space obtained by t-SNE dimensionality reduction, and there is no actual attribute and unit. Each point represents only the coordinate position of the sample, and different colors correspond to different classes [37]. Therefore, we do not mark the attributes and units on the horizontal and vertical axes in Fig. 5.

A careful look at Fig. 5(a) reveals that the original features of different classes are too intensive and overlapping, it difficult to distinguish between different classes. Similarity as shown in Fig. 5(c), the features of classes 8, 9, and 10 are randomly scattered in other class spaces, and there is a large overlap between different classes, which can cause misclassification. However, it can be observed in Fig. 5(b), features between the same classes are relatively aggregated and separated from each other. This indicates that NCAE can be used to extract more distinctive hidden features of the driving behavior and aiding the classifier to map out the separating boundaries among the classes more easily.

3.6. Supervised learning: Driver identification

In the last section, a series of driver identification experiments are used to evaluate the effect of the proposed NCAE network on classification. We construct a DNCAE for driver classification. The DNCAE contains six stacked NCAEs and a softmax classifier. we build the DNCAE structure for the 1530-765-380-190-85-40-10 configuration.

We first used the training sample set to conduct unsupervised pretraining on six NCAEs to learn the weights. Then, we stacked six NCAEs to output the hidden activities to the softmax classifier for the supervised training for classification. In addition, to further improve the classification accuracy, we fine-tuned the pretrained DNCAE to correct the network parameters. Finally, the trained DNCAE was used to evaluate the classification performance of the proposed method on the test sample set. We choose the classification accuracy of the proposed method on the test sample set and training sample set as an evaluation index. The formulas are as follows:

$$A_{train} = \frac{Number\ of\ correctly\ classified\ train\ cases}{Total\ number\ of\ train\ cases} \tag{19}$$

$$A_{test} = \frac{Number\ of\ correctly\ classified\ test\ cases}{Total\ number\ of\ test\ cases} \tag{20}$$

In the first experiment, based on previous researches [32,33, 35,36,38,39], we use the different network architectures of Sparse Autoencoder (SAE) (SAE, that is, adding KL constraint and $L2$ weight regularization based on autoencoder) [32,33,38], Dropout-Autoencoder (DpAE) [39], and NCAE for the classification experiments on the driving behavior dataset. Table 1 shows that the

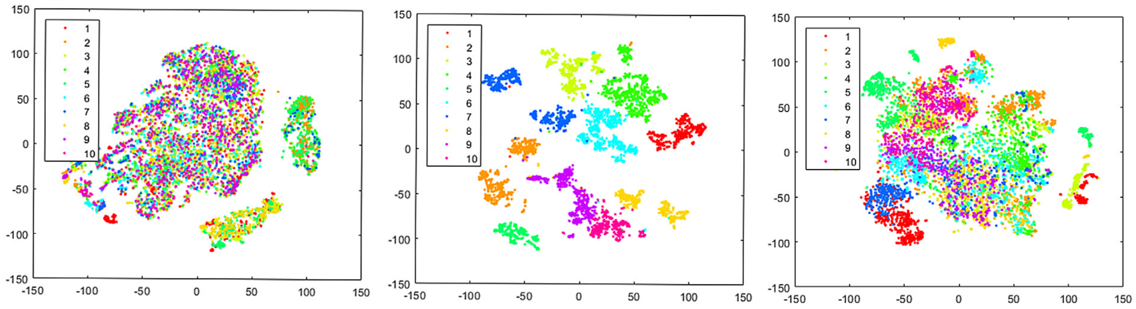


Fig. 5. t-SNE projection of high-level representations of driving behavior features using (a) 1530D Normalized raw windowing data, (b) 10D representations using DNCAE, and (c) 360D representations using reference methods (PCA) [7,8].

classification results are averaged over ten independent experiments to mitigate the effect of initialization and the related mean values and SDs of three architectures. The results indicate that relative to the conventional SAE and DpAE, the proposed NCAE architecture achieves higher classification accuracy both before and after fine-tuning the test sample set, reaching 90.79% $\pm$ 0.76% and 99.65% $\pm$ 0.22%, respectively.

In addition, to verify the effect of each architecture on overfitting, we calculate the difference of the classification accuracy of each architecture on the training sample set and the test sample set. Table 1 shows that when using the NCAE architecture, the A_{test}-A_{train} is only 0.34% $\pm$ 0.22%, indicating that the proposed NCAE architecture for driver classification offers better performance on reducing overfitting and the generalization ability of the model.

Simultaneously, the convergence speeds of the different architectures are compared based on the number of iterations required to achieve convergence during fine-tuning. These speeds are listed in the last column of Table 1. The NCAE architecture require fewer iterations than other architectures, indicating that NCAE can achieve faster convergence speed without the additional cost of more iterations while ensuring better classification accuracy.

In the second experiment, we compare the classification effect of our method and the current methods [7,8,15–18] for driver classification on the driving behavior dataset. Table 2 reports the average classification results for ten experiments of each algorithm. By comparing the conventional algorithms [7,8], we find that the NCAE algorithm achieves higher classification accuracy on the test sample set. In addition, we calculate the time required for each algorithm to predict the class of a test sample. It can be found that the proposed method guarantees the highest prediction accuracy, while having the shortest prediction time and better timeliness.

4. Conclusions and future directions

We proposed a new driver identification method based on deep learning. Firstly, we have presented a new unsupervised learning method to adaptive search the optimal size of the sliding time window. Then, we construct a DNCAE network to extract the hidden features of driving behavior and further realize driver identification. The performance of the proposed method was compared with that of the conventional SAE and DpAE, and the existing Random Tree and Random Forest methods for driver classification. We evaluated the performance of the proposed method on the public driving behavior dataset.

In the experimental part of verifying the adaptive search the optimal size of the sliding window, we build a three-layer NCAE network for unsupervised learning and automatically update the window size in the process of minimizing the objective function to find the optimal size value 30 s. The comparative results show that using 30 s as the optimal window size, it can effectively reduce the reconstruction error, and better preserve the intrinsic information of the original data.

In the supervised classification experimental stage, the cross-validation experimental results indicate that the proposed DNCAE model can obtain relatively high classification accuracy and shorter prediction time. In addition, our method can more effectively reduce overfitting and improve the generalization ability of the model.

In the future, we need to further work from the following aspects:

(1) Due to the privacy protection of some natural driving datasets, most datasets do not disclose the complete driving behavior time series data corresponding to different drivers. Therefore, we only use a public driving behavior data for verification experiments in this paper. In the future, we need to find more large-scale Naturalistic Driving Studies (NDS) datasets, such as 100-CAR [40–42], SHRP2 NDS dataset [43–45], and further expand the network architecture and depth to build a deeper NCAE model for verification.

(2) The values of some key hyperparameters in our proposed DNCAE model are the optimal values selected according to our experience and experimental results (Discussed in Section 3.2). In the future, we can use some optimizers to adaptively search for parameters values to improve the applicability of the model [46–50].

(3) Our model still has some shortcomings that need further study. For example, as our model architecture grows, it will lead to two problems. First, As the number of nodes in the hidden layers increases, the number of weight parameters to be learned exponentially increases, which causes an important challenge in which the network parameters tend to overfit to the given training data [51]. How to compress the network mass, reduce the number of parameters, improve the efficiency and generalization ability of the model, reduce over-fitting. This is a research issue that needs attention in the future. On the other hand, Recent studies indicate that modern deep networks often have large number of overlapping filters amounting to unnecessary filtering redundancy and large inference cost [35,52,53], that is, some filters are repeated. In the future, we will use clustering or constrained network cost function to eliminate feature redundancy.

(4) In addition, due to many sudden factors in the driving process, even the same driver, there is a certain degree of difference in driving behavior in different road conditions and at different times. In the future, we can consider adding some random noise to the original time series data of driving behavior to train our proposed model to improve the generalization ability of the model.

(5) We can use our model to solve more practical problems. For example, in Advanced Driver Assistance System (ADAS) [54,55], our model can be used to real-time monitor driving behavior and immediately warn of driving behavior that may result in high risk.

Table 1
Classification performance on the driving behavior dataset using different network architectures.

Architecture	Before fine-tuning	After fine-tuning			
	A_{test}	A_{train}	A_{test}	A_{test} -A_{train}	Iterations
SAE	26.43 ± 2.72	98.86 ± 0.08	93.80 ± 0.06	5.06 ± 0.06	400
NCAE	90.79 ± 0.76	99.99 ± 0.01	99.65 ± 0.22	0.34 ± 0.22	85
DpAE (20% dropout)	65.32 ± 1.08	95.34 ± 0.17	92.25 ± 0.04	3.09 ± 0.09	400

Table 2
Comparing the classification performance with existing methods.

Feature set	Feature dimension	Size of sliding window Δt (s)	Method	Classification accuracy A_{test}	Time (s)
Hidden features using DNCAE (Section 3.3(A))	10D	30	DNCAE+SMC	99.65 ± 0.22	2.77e−5
Features using PCA [7]	360D	60	PCA + RandomTree	98.40 ± 0.14	6.21e−3
Original features and statistical features [8]	945D	60	InfoGainAttributeEval+Random Forest	96.72 ± 0.83	2.12e−2

In addition, in UBI [11,12,41], how to accurately assess the risk level of driving behavior is a key factor in auto insurance pricing. In response to the above problems, our model may have certain advantages in feature extraction and representation.

References

[1] U. Fugiglando, E. Massaro, P. Santi, S. Milardo, K. Abida, R. Stahlmann, F. Netter, C. Ratti, Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment, (2017), arXiv:1710.04133.
[2] A. Bouhoute, I. Berrada, M.E. Kamili, A formal model of human driving behavior in vehicular networks, in: Wireless Communications and Mobile Computing Conference, 2014, pp. 231-236.
[3] F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, Car hacking identification through fuzzy logic algorithms, in: IEEE International Conference on Fuzzy Systems, 2017, pp. 1-7.
[4] K.M.A. Alheeti, A. Gruebler, K.D. Mcdonald-Maier, An intrusion detection system against malicious attacks on the communication network of driverless cars, in: 12th Annual IEEE Consumer Communications and Networking Conference, 2015, pp. 916-921.
[5] N. Lyamin, A. Vinel, M. Jonsson, J. Loo, Real-Time detection of denial-of-service attacks in IEEE 802.11p Vehicular Networks, IEEE Commun. Lett. 18 (2014) 110–113.
[6] A. Taylor, S. Leblanc, N. Japkowicz, Anomaly detection in automobile control network data with long short-term memory networks, in: IEEE International Conference on Data Science and Advanced Analytics, 2016, pp. 130-139.
[7] F. Martinelli, F. Mercaldo, A. Orlando, V. Nardone, A. Santone, A.K. Sangaiah, Human behavior characterization for driving style recognition in vehicle system, Comput. Electr. Eng. 000 (2018) 1–16.
[8] B.I. Kwak, J.Y. Woo, H.K. Kim, Know your master: Driver profiling-based anti-theft method, in: Privacy, Security and Trust, 2017, pp. 211-218.
[9] A. Marotta, F. Martinelli, S. Nanni, A. Orlando, A. Yautsiukhin, Cyber-insurance survey ☆, Comput. Sci. Rev. 24 (2017) 35–61.
[10] D.I. Tselentis, G. Yannis, E.I. Vlahogianni, Innovative motor insurance schemes: A review of current practices and emerging challenges, Accid. Anal. Prev. 98 (2016) 139.
[11] Y. Bian, C. Yang, J.L. Zhao, L. Liang, Good drivers pay less: A study of usage-based vehicle insurance models, Transp. Res. A 107 (2018) 20–34.
[12] J.S. Wijnands, J. Thompson, G.D.P.A. Aschwanden, M. Stevenson, Identifying behavioural change among drivers using long short-term memory recurrent neural networks, Transp. Res. F 53 (2018) 34–49.
[13] E. Massaro, C. Ahn, C. Ratti, P. Santi, R. Stahlmann, A. Lamprecht, M. Roehder, M. Huber, The car as an ambient sensing platform, Proc. IEEE 105 (2016) 3–7.
[14] A. Santini, OBD-II: Functions, Monitors and Diagnostic Techniques, Delmar, (2010).
[15] M. Enev, A. Takakuwa, K. Koscher, T. Kohno, Automobile driver fingerprinting, Proc. Privacy Enhanc. Technol. 2016 (2016) 34–50.
[16] G. Kedar-Dongarkar, M. Das, Driver classification for optimization of energy usage in a vehicle, Procedia Comput. Sci. 8 (2012) 388–393.
[17] M.V. Ly, S. Martin, M.M. Trivedi, Driver classification and driving style recognition using inertial sensors, Intell. Veh. Symp. 36 (2013) 1040–1045.
[18] X. Zhang, X. Zhao, J. Rong, A study of individual characteristics of driving behavior based on hidden markov model, Sens. Transducers 167 (2014) 194–202.
[19] A. Wahab, C. Quek, C.K. Tan, K. Takeda, Driving profile modeling and recognition based on soft computing approach, IEEE Trans. Neural. Netw. 20 (2009) 563–582.
[20] S. Choi, J. Kim, D. Kwak, P. Angkititrakul, J.H.L. Hansen, Analysis and classification of driver behavior using in-vehicle CAN-BUS Information, in: Biennial Workshop on DSP for In-Vehicle and Mobile Systems, 2007, pp. 17–19.
[21] E. Hosseini-Asl, J.M. Zurada, O. Nasraoui, Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints, IEEE Trans. Neural. Netw. Learn. Syst. 27 (2016) 2486–2498.
[22] J. Chorowski, J.M. Zurada, Learning understandable neural networks with nonnegative weight constraints, IEEE Trans. Neural. Netw. Learn. Syst. 26 (2015) 62–69.
[23] B.O. Ayinde, E. Hosseini-Asl, J.M. Zurada, Visualizing and understanding non-negativity constrained sparse autoencoder in deep learning, in: International Conference on Artificial Intelligence and Soft Computing, 2016, pp. 3-14.
[24] A. Lemme, R.F. Reinhart, J.J. Steil, Online learning and generalization of parts-based image representations by non-negative sparse autoencoders, Neural Netw. 33 (2012) 194–203.
[25] J. Xu, L. Xiang, R. Hang, J. Wu, Stacked Sparse Autoencoder (SSAE) based framework for nuclei patch classification on breast cancer histopathology, in: IEEE International Symposium on Biomedical Imaging, 2014, pp. 999-1002.
[26] H. Ohno, Linear guided autoencoder: representation learning with linearity, Appl. Soft Comput. 55 (2017) 566–575.
[27] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: International Conference on Machine Learning, 2008, pp. 1096-1103.
[28] G.E. Hinton, R.S. Zemel, Autoencoders, minimum description length and Helmholtz free energy, in: International Conference on Neural Information Processing Systems, 1993, pp.3-10..
[29] T.D. Nguyen, T. Tran, D. Phung, S. Venkatesh, Learning parts-based representations with nonnegative restricted boltzmann machine, in: Asian Conference on Machine Learning, 2013, pp. 133-148.
[30] H. Lee, C. Ekanadham, A.Y. Ng, Sparse deep belief net model for visual area V2, in: Advances in neural information processing systems, 2008, pp. 873-880.
[31] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) 1527–1554.
[32] E. Hosseini-Asl, J.M. Zurada, O. Nasraoui, Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints, IEEE Trans. Neural. Netw. Learn. Syst. 27 (2016) 2486–2498.
[33] H.L. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, T. Bando, Visualization of driving behavior based on hidden feature extraction by using deep learning, IEEE Trans. Intell. Transp. Syst. 18 (2017) 2477–2489.
[34] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (1995) 1190–1208.
[35] B.O. Ayinde, J.M. Zurada, Nonredundant sparse feature extraction using autoencoders with receptive fields clustering, Neural Netw. 93 (2017) 99–109.
[36] B.O. Ayinde, J.M. Zurada, Clustering of receptive fields in autoencoders, in: Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 1310–1317.
[37] L.V.D. Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[38] H. Liu, T. Taniguchi, T. Takano, Y. Tanaka, Visualization of driving behavior using deep sparse autoencoder, in: Intelligent Vehicles Symposium Proceedings, 2014, pp. 1427-1434.
[39] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, Comput. Sci. 4 (2012) 212–223.
[40] S.G. Klauer, T.A. Dingus, T.V. Neale, J. Sudweeks, D.J. Ramsey, The Impact of Driver Inattention on Near-Crash/Crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study Data, U.s.department of Transportation Washington D.c, 226 (2006), https://doi.org/DOTHS810594.
[41] F. Guo, Y. Fang, Individual driver risk assessment using naturalistic driving data, Accid Anal. Prev. 61 (2013) 3–9.
[42] H.S. Dot, An analysis of driver inattention using a case-crossover approach on 100car data: final report, Drowsiness 148 (2010) 1–148.
[43] J. Antin, S. Lee, J. Hankey, T. Dingus, Design of the In-Vehicle Driving Behavior and Crash Risk Study: In Support of the SHRP 2 Naturalistic Driving Study, Shrp Report, (2011), https://doi.org/10.17226/14494.

[44] K.L. Campbell, The SHRP 2 naturalistic driving study: Addressing driver performance and behavior in traffic safety, Trans. News 282 (2012) 30–35.
[45] J. Bärgman, V. Lisovskaja, T. Victor, C. Flannagan, M. Dozza, How does glance behavior influence crash and injury risk? A 'what-if' counterfactual simulation using crashes and near-crashes from SHRP2, Transp. Res. F 35 (2015) 152–169.
[46] H. Faris, M.M. Mafarja, A.A. Heidari, I. Aljarah, A.M. Al-Zoubi, S. Mirjalili, H. Fujita, An efficient binary salp swarm algorithm with crossover scheme for feature selection problems, Knowl.-Based Syst. 154 (2018) 43–67.
[47] A.A. Heidari, R.A. Abbaspour, A.R. Jordehi, An efficient chaotic water cycle algorithm for optimization tasks, Neural Comput. Appl. 28 (2017) 57–85.
[48] A.A. Heidari, P. Pahlavani, An efficient modified grey wolf optimizer with lévy flight for optimization tasks, Applied Soft Computing 60 (2017) 115–134.
[49] M. Mafarja, I. Aljarah, A.A. Heidari, A.I. Hammouri, H. Faris, A.M. Al-Zoubi, S. Mirjalili, Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems, Knowl.-Based Syst. 145 (2017) 25–45.
[50] A.A. Heidari, R.A. Abbaspour, A.R. Jordehi, Gaussian bare-bones water cycle algorithm for optimal reactive power dispatch in electrical power systems, Appl. Soft Comput. 57 (2017) 657–671.
[51] A. Sankaran, M. Vatsa, R. Singh, A. Majumdar, Group sparse autoencoder, Image Vis. Comput. 60 (2017) 64–74.
[52] B.O. Ayinde, J.M. Zurada, Building efficient convnets using redundant feature pruning, in: ICLR, (2018).
[53] J. Chen, Z. Wu, J. Zhang, F. Li, W. Li, Z. Wu, Cross-covariance regularized autoencoders for nonredundant sparse feature representation, Neurocomputing 316 (2018) 49–58.
[54] V.A. Butakov, P. Ioannou, Personalized driver/vehicle lane change models for ADAS, IEEE Trans. Veh. Technol. 64 (2015) 4422–4431.
[55] P. Angkititrakul, R. Terashima, T. Wakita, On the use of stochastic driver behavior model in lane departure warning, IEEE Trans. Intell. Transp. Syst. 12 (2011) 174–183.