Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

CNN and KPCA Based Automated Feature Extraction for Real Time Driving Pattern Recognition

LIANG XIE[1], JILI TAO[2], QIANNI ZHANG[3] AND HUIYU ZHOU.[4]

[1]College of Control Science and Engineering,Zhejiang University, Hangzhou, CO 310027 P.R.China (e-mail: xieliang65@zju.edu.cn)
[2]Ningbo Institute of Technology, Zhejiang University, Ningbo, CO 315100 P.R.China (e-mail: tjl810@126.com)
[3]School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom. (e-mail:qianni.zhang@eecs.qmul.ac.uk)
[4]Informatics, University of Leicester, LE1 7RH,United Kingdom. (e-mail: hz143@leicester.ac.uk)

Corresponding author: Jili Tao (e-mail: tjl810@126.com).

This work was supported in part by the National Natural Science Foundation of China under Grant 61603337 and the Zhejiang province natural science fund under Grant LY19F030009. H. Zhou was supported by UK EPSRC under Grant EP/N011074/1 and Royal Society-Newton Advanced Fellowship under Grant NA160342, and European Union's Horizon 2020 Research and Innovation Program through the Marie-Sklodowska-Curie under Grant 720325.

ABSTRACT Driving conditions greatly affect the energy control and the fuel economy of a hybrid electric vehicle (HEV). In this paper, an automated feature extraction scheme based on convolution neural networks (CNNs) and Kernel PCA (KPCA) for real time driving pattern recognition (RTDPR) is proposed in order to achieve consistent performance of the energy management. Firstly, a dimension expanding strategy is performed to transform one-dimensional speed sequences to generate a two-dimensional dataset. Then, the transformed data is sent to the CNN and KPCA based feature extractor. Finally, the feature extractor automatically selects the most representative features for classification. To improve the generalization of CNN to a small sample dataset, the structure of the typical CNN is adjusted by adding the KPCA layer in order to reduce model parameters. The model is well trained and evaluated in simulation, and it is tested for RTDPR in the real world. Simulation and experimental results show that the proposed automated feature extraction strategy outperforms the conventional driving pattern recognition algorithms based on manually feature extraction, which has achieved the state-of-the-art recognition accuracy.

INDEX TERMS Convolution neural network, driving pattern recognition, feature selection, kernel principle component analysis.

I. INTRODUCTION

A driving pattern is typically defined as the driving cycle of a vehicle in a particular environment [1], [2]. Since the current driving pattern has a great impact on the energy management strategy of a hybrid electric vehicle (HEV) [3], [4], it is efficient to use the prior knowledge of the driving cycle to achieve the real time driving pattern recognition (RTDPR) and enhance the control performance of the HEV [5], [6]. There are many researches on the RTDPR [7]–[11]. The conventional way is to manually extract features from the historical speed data to characterize the driving patterns [7]. Then the classical machine learning models like k-means [8], hidden Markov models [9], fuzzy c-means [10], and their variants [11] are fully utilized to classify the extracted features into different categories. Therefore, the quality of the feature extraction algorithm plays a great impact on the classification accuracy. However, those manually extracted features usually include average speeds, average accelerations and other features which are directly calculated using physical models [12], while other complex and high level features are hard to represent. In practice, those low level features are unable to effectively characterize the complex driving patterns. Additionally, to reduce time cost of RTDPR, a limited amount e.g., 16 features are selected to characterize the driving patterns [13]. Based on the above analysis, it can be concluded that the recognition accuracy of the conventional methods is significantly affected by selected features. Recently with the development of deep learning and its strong classification ability [14]–[16], the convolution neuron network (CNN) has been wildly used in the pattern recognition fields [17]–[19], and achieved good performances. The CNN can achieve an end-to-end recognition without feature

extraction but still has not been widely applied in RTDPR, partially due to the lack of magnanimous training samples. Motivated by the CNN, we do not manually generate the feature vectors from the historical speed data to build the model. Instead, the model learns to extract the features itself from the datasets [20]. During the training process, the model can learn to select the most representative features and their amount automatically. The simulation results indicate that the features selected automatically by the models are more representative than those that are manually designed. The standard CNN is a nonlinear model with typically thousands of parameters, which may easily get overfitting when the training samples are not sufficient [21]. The most parameters concentrate on the fully-connected layers which hold much redundancy. To solve the problem, we design an automated feature extractor that retains the former part of the CNN and removes the fully-connected layer. Then the kernel PCA (KPCA) layer is added to further supply features, thus the redundancy is removed and classification is simplified. Additionally, we have performed linear shift on the speed data to expand the dataset, which also proves to be very effective to avoid overfitting.

In this work, we firstly collect the training samples from the historical speed data by a sliding window. The size and step of the window are adjusted in the training process. Secondly, we transform the training samples to the two-dimension dataset so that the CNN based model can effectively deal with the speed information. Thirdly, the two-dimension dataset are divided into batches to fit the feature extractor. Finally, the extracted features are utilized for RTDPR. The specific contributions of this paper are as follows: (1) We have improved the generalization of the standard CNN for small dataset by adding the KPCA layer. (2) We have achieved an end-to-end strategy for RTDPR instead of manually designing features. (3) The historical speed sequence is transformed to two-dimension to extract spatial features. (4) We have achieved the state-of-the-art accuracy for RTDPR.

The structure of this paper is as follows: the details of the CNN+KPCA architecture are described in section II. Then our model based on CNN+KPCA is reported in section III. Section IV presents the applications on four typical patterns in the congested urban, flowing urban, subway and high way and in real environment. The results are compared with that of other typical classifiers. Finally, section V gives the conclusions of this paper.

II. THE CNN+KPCA ARCHITECTURE

A. THE STANDARD CNN CLASSIFIER

The CNN model is a complex nonlinear function that maps the input samples into the corresponding driving patterns. The overall structure of CNN is described in Fig. 1, which includes one input layer, the complex middle layers and one output layer. The input layer of the CNN deals with the two-dimension samples. The middle layers include the convolution layers and a fully-connected layer. Within the convolution layer, the convolution operation is performed, followed by the max-pooling operation immediately. The outputs of the last convolution layer are then flattened to one-dimension as the inputs of the fully connected layer for further nonlinearization. In the output layer, there contain four neurons that delegate different driving patterns. The details of the calculation process are described as follows.

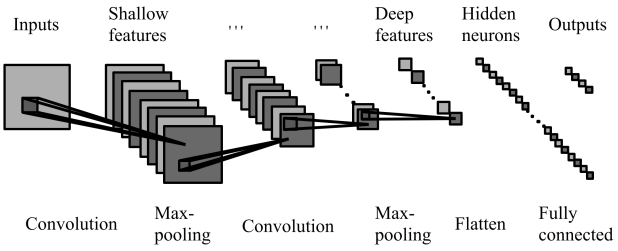


FIGURE 1: The architecture of typical CNN classifier

Provided that we have an $n \times n$ input sample represented as a two-dimension array:

$$\boldsymbol{x} = \{x_{0,0}...x_{0,n-1}; x_{1,0}...x_{1,n-1}; x_{n-1,0}...x_{n-1,n-1}\} \tag{1}$$

where $x_{i,j}$ is the pixel at the position (i, j). A two-dimension kernel is defined to connect the input layer to the convolution layer, which is represented by a $m \times m$ array of shared weights:

$$\boldsymbol{w} = \{w_{0,0}...w_{0,m-1}; w_{1,0}...w_{1,m-1}; w_{m-1,0}...w_{m-1,m-1}\}. \tag{2}$$

A two-dimension feature map is then obtained as described in (3).

$$C^1_{i,j,k} = f(b_k + \sum_{a_1=0}^{m-1} \sum_{a_2=0}^{m-1} w_{a_1 a_2} x_{i+a_1, j+a_2}) \tag{3}$$

Here, $C^1_{i,j,k}$ represents the i, j-th value of the feature map to the k-th kernel in the first convolution operation. f is the activation function of the neuron, which is described as:

$$f(x) = max(0, x). \tag{4}$$

b_k is the shared bias of the k-th kernel. Finally, we use $x_{i+a_1, j+a_2}$ to denote the pixel at the position $(i+a_1, j+a_2)$. The shared weights and bias of the kernel means that the same features will be detected, just at different locations of the input sample. Different kernels generate different feature maps, which comprises the former part of the convolution layer.

The max-pooling operation prepares the condensed feature maps from the former part of the convolution layer. Then those feature maps stack up and comprise the latter part of the convolution layer. For instance, each neuron in the pooling operation summarizes a maximum activation in a region of

$e \times e$ neurons in the feature maps. For the i, j, k-th (zero-base) neuron in the first max-pooling operation, the output is:

$$M^1_{i,j,k} = \max\{C^1_{ei+\sigma_1, ej+\sigma_2, k} \mid 0 \le \sigma_1, \sigma_2 < e\}. \quad (5)$$

The second convolution layer is obtained by performing the convolution operation $\boldsymbol{C^2}$ and the max-pooling operation $\boldsymbol{M^2}$. The details of those operations are described in (3) and (5), respectively. Then $\boldsymbol{M^2}$ is flattened from three-dimension to one dimension, i.e., neurons in this layer are arranged in line. The i-th activation in the flattened layer is represented as M^2_i.

Finally, the fully connected layer connects every neuron from the flattened max-pooling layer to every one of 4 output neurons [22]. Assume that there are h neurons in the fully connected layer, then the i-th activations of the fully connected layer (F_i) and the output layer (Y_i) are described in (6) and (7), respectively.

$$F_i = f(\varphi^1_i + \sum_{j=0}^{h-1} \theta^1_{ji} M^2_j),\ i \in [0, h) \quad (6)$$

$$Y_i = s(\varphi^2_i + \sum_{j=0}^{h-1} \theta^2_{ji} F_i),\ i \in [0, 4) \quad (7)$$

where $\boldsymbol{\varphi^1}$ and $\boldsymbol{\theta^1}$ are the biases and weights between the flattened max-pooling layer and the fully connected layer, respectively. $\boldsymbol{\varphi^2}$ and $\boldsymbol{\theta^2}$ are the biases and weights between the fully connected layer and the output layer, respectively. s is the softmax function described in (8).

$$s(z)_j = \frac{e^{z_j}}{\sum_{i=0}^{3} e^{z_i}},\ j \in [0, 4) \quad (8)$$

B. THE CNN+KPCA FEATURE EXTRACTOR

As shown in Fig. 2, the number of the parameters in the first convolution layer is $m \times m \times k^1$, in the second convolution layer is $m \times m \times k^1 \times k^2$, where k^1, k^2 are the number of the kernels in the first and second convolution layer respectively. In the flatten layer, the parameter number is $(\frac{n}{4})^2 \times k^2$, which is thousand magnitude due to the width of the input n. The default number of the hidden neurons in the post-fully-connected layer is always set to be 1024 or 2048 [23]. Thus the parameter number in the fully-connected layer can reach a million level, which can easily result in overfitting in the task of small samples classification. We propose two methods to reduce the parameters while keeping the CNN performance. Firstly, we remove the fully-connected layer which contains large redundancy. Secondly, we retain the convolution layer, the output of which are projected by the KPCA operation, i.e. the KPCA layer. The dimension of the flattened convolution layer can be reduced considerably from thousands to ten by KPCA projection. While the major feature information is retained. The combined convolution and KPCA layers are treated as the feature extractor as described in Fig. 2. The output of the flattened convolution layer is calculated by Eq.(1)-(4), represented as $\boldsymbol{M^2}$. The KPCA projection is performed to reduce the feature dimension calculated as follows.

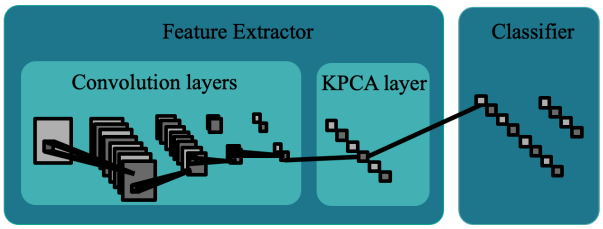


FIGURE 2: The structure of the CNN+KPCA feature extractor

$$K_{ij} = k(M^2_i, M^2_j) \quad (9)$$

where k is the kernel function, K_{ij} represents the i, j-th element of the kernel matrix. Then the kernel matrix is centralized by (10).

$$\overline{K} = K - 1_N K - K 1_N + 1_N K 1_N \quad (10)$$

Where $1_N \in R^{N \times N}, (1_N)_{i,j} = \frac{1}{N}$, N is the number of the training samples. Calculating the eigen value λ_i and the corresponding eigen vector a_i of $\overline{K}$, and perform normalization to the eigen vector as follows.

$$a_i \leftarrow \frac{a_i}{\sqrt{\lambda_i}} \quad (11)$$

Then select the corresponding normalized eigen vectors according to the former maximum λ. The number of the selected eigen vectors l is decided by the accumulated contribution rate calculated as follows.

$$E = \frac{\sum_{i=1}^{l} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \quad (12)$$

At last the feature dimension is reduced by (13).

$$\boldsymbol{M^2_{kpca}} = \overline{K}\boldsymbol{\alpha} \quad (13)$$

where $\boldsymbol{\alpha} = [a_1, a_2 ... a_l]$, $\boldsymbol{M^2_{kpca}}$ are the extracted features after KPCA layer. The extracted features are then utilized for the post classification.

Provided that we have a speed sequence of a driving vehicle with an unknown pattern, our goal is to recognize the driving pattern. The CNN+KPCA model is applied as a classifier to provide the probability distribution of the driving patterns for each input sequence, which is illustrated in Fig. 3. Each neuron of the output layer represents a driving pattern. The driving pattern with the maximum probability is the final recognized result.

III. MODEL BUILDING

Since the KPCA fitting requires to maintain the projection space unchanged, the model building will include two separate processes. In the first stage, we train the standard CNN model with stochastic gradient descent methods. After the

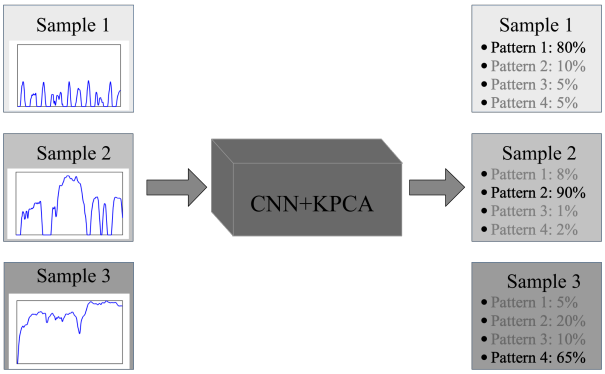


FIGURE 3: Driving pattern recognition process

parameters of the CNN are optimized, we add the KPCA layer between the convolution layer and the fully-connected layer of the CNN. And in the second stage, we fine-tune the fully-connected layer with the convolution layer frozen. The parameters of the fully-connected layer are updated iteratively by gradient descent with the whole training batch. The computation is described in (14)-(20) in detail.

A. TYPICAL CNN MODEL BUILDING

We use the categorical cross-entropy in the loss function for the multi-class classification problem [24], which is detailed in (14).

$$L(\boldsymbol{z}) = -\sum_{i=0}^{3} G_i log(Y_i) \tag{14}$$

where G is the ground truth of the sample. $\boldsymbol{z}$ represents the parameters to be optimized, i.e., $\boldsymbol{z} = [\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\theta^1}, \boldsymbol{\theta^2}, \boldsymbol{\varphi^1}, \boldsymbol{\varphi^2}]$. All the operations on the vectors are element-wise. An algorithm for the first-order gradient based optimization of the stochastic objective functions is used to minimize these parameters. The adaptive estimation of the lower-order moments [25] is as follows:

$$\boldsymbol{g}_{\boldsymbol{z},t} = \nabla_{\boldsymbol{z}} L(\boldsymbol{z}_t) \tag{15}$$

where $\boldsymbol{g}_{\boldsymbol{z},t}$ is the gradients of $\boldsymbol{z}$ at iteration t. Denoting $\boldsymbol{m}_{\boldsymbol{z},t}$ as the exponential moving averages of the gradient $\boldsymbol{g}_{\boldsymbol{z},t}$, $\boldsymbol{v}_{\boldsymbol{z},t}$ as the exponential moving averages of the squared gradient $\boldsymbol{g}^2_{\boldsymbol{z},t}$, then a momentum term is introduced to update the values of $\boldsymbol{m}_{\boldsymbol{z},t}$ and $\boldsymbol{v}_{\boldsymbol{z},t}$ as described in (16) and (17),

$$\boldsymbol{m}_{\boldsymbol{z},t} = \beta_1 \boldsymbol{m}_{\boldsymbol{z},t-1} + (1-\beta_1)\boldsymbol{g}_{\boldsymbol{z},t} \tag{16}$$

$$\boldsymbol{v}_{\boldsymbol{z},t} = \beta_2 \boldsymbol{v}_{\boldsymbol{z},t-1} + (1-\beta_2)\boldsymbol{g}^2_{\boldsymbol{z},t} \tag{17}$$

where $\beta_1, \beta_2 \in [0, 1)$ are two hyper parameters that control the exponential decay rate of the gradient and the squared gradient, respectively. When the moment estimations are initialized as zero or the βs are initialized close to zero, those estimations will be biased toward zero. To avoid this, $\boldsymbol{m}_{\boldsymbol{z},t}$ and $\boldsymbol{v}_{\boldsymbol{z},t}$ are bias-corrected as follows:

$$\overline{\boldsymbol{m}}_{\boldsymbol{z},t} = \frac{\boldsymbol{m}_{\boldsymbol{z},t}}{1-\beta_1} \tag{18}$$

$$\overline{\boldsymbol{v}}_{\boldsymbol{z},t} = \frac{\boldsymbol{v}_{\boldsymbol{z},t}}{1-\beta_2} \tag{19}$$

where $\overline{\boldsymbol{m}}_{\boldsymbol{z},t}$ and $\overline{\boldsymbol{v}}_{\boldsymbol{z},t}$ are the bias-corrected first moment estimates and bias-corrected second moment estimates, respectively. Finally, the bias-corrected estimates are used to update the parameters (20),

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \alpha \frac{\overline{\boldsymbol{m}}_{\boldsymbol{z},t}}{\sqrt{\overline{\boldsymbol{v}}_{\boldsymbol{z},t}} + \epsilon} \tag{20}$$

where α and ϵ represents the learning rate and the target error, respectively.

It is worthy to mention that the learning process in the two different training phases is very similar. There only exist slight differences of the objective parameters.

B. CNN+KPCA MODEL BUILDING

After we have built the standard CNN, we extract the convolution layer and combine it with the KPCA layer as the feature extractor. Afterwards, we obtain all the features from the dataset which are used to develop the classifier. The training process is the same as the standard CNN described in (14)-(20), leaving a different input and the objective parameters to optimize. The feature extractor remains the same in this developing phrase, and the parameters in (21) are optimized following the way described in (14)-(20).

$$\boldsymbol{z} = [\boldsymbol{\theta^1}, \boldsymbol{\theta^2}, \boldsymbol{\varphi^1}, \boldsymbol{\varphi^2}] \tag{21}$$

IV. CASE STUDY

A. TYPICAL DRIVING PATTERN

The speed-time sequence under different driving conditions is sampled to implement the driving pattern recognition. The Environmental Protection Agency (EPA) has classified four typical driving patterns in the real world, which include congested urban roads, flowing urban roads, subway and highway. The corresponding driving conditions are Manhattan bus drive cycle (MBDC), EPA urban dynamometer driving schedule (UDDS), West Virginia suburban driving schedule (WVUSUB) and US EPA highway fuel economy certification test (HWFET), which are labeled from 1 to 4 respectively indicating 4 different groups. The characteristics of the four driving conditions are described in Table 1, and the speed-time sequences are shown in Fig. 4.

TABLE 1: Four Typical Driving Conditions

Driving Conditions	Label	Characteristics
MBDC	1	represents low speed go-and-stop traffic driving condition.
UDDS	2	represents city driving condition.
WVUSUB	3	represents suburban driving condition.
HWFET	4	represents highway driving condition under 60mph

B. THE DATASET PROCESS

To expand the dataset, a sliding window is defined as shown in Fig. 5, where the $width$ represents the width of the window

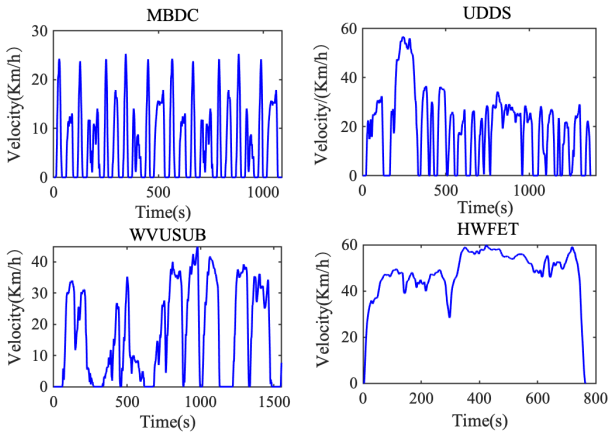


FIGURE 4: The speed-time sequences of four typical driving conditions

and the $step$ represents the sliding step of the window. The $width$ controls the size of the samples. The bigger the $width$ is, the more information is contained in the samples but it takes more time to acquire the samples. The $step$ controls the size of the dataset. The smaller the $step$ is, the more samples are collected from the dataset but the samples become more similar to each other. Therefore, the $width$ and $step$ are two variables that need to be selected carefully. In this work, $width$ and $step$ are experimentally set as 40 and 5 seconds respectively. To collect samples, the window slides through the speed-time sequence. At each step it slides through, a sequence that represents the corresponding driving pattern is obtained, e.g., the first step of the window is [0,40], the second is [5,45], and the third is [10,50], etc. Then the corresponding ground truth is labeled with the sequence. We use a four-bit binary code to represent the ground truth (e.g., 0001 for label 1 and 0010 for label 2).

In the one-dimension sequence, only the speed information is taken into consideration. To exploit the spatial structure of the speed distribution, we take a close look at the sequences on the pixel-wise level. A two-dimension array is used to reconstruct the sequence as shown in Fig. 5, where the 1s represent the corresponding pixels that contain the speed information in the 1-dimension sequence, and the 0s represent the corresponding pixels without the speed information in the 1-dimension sequence. The resolution of the 2-dimension data is defined by $n \times n$, which controls the information amount of the transformed samples. The transformed samples with their labels are used to optimize the parameters of the model.

C. HYPER PARAMETERS

Table 2 gives the hyper parameters of the typical benchmark models and the proposed CNN+KPCA model, respectively, where k^1 and k^2 represents the number of the kernels in

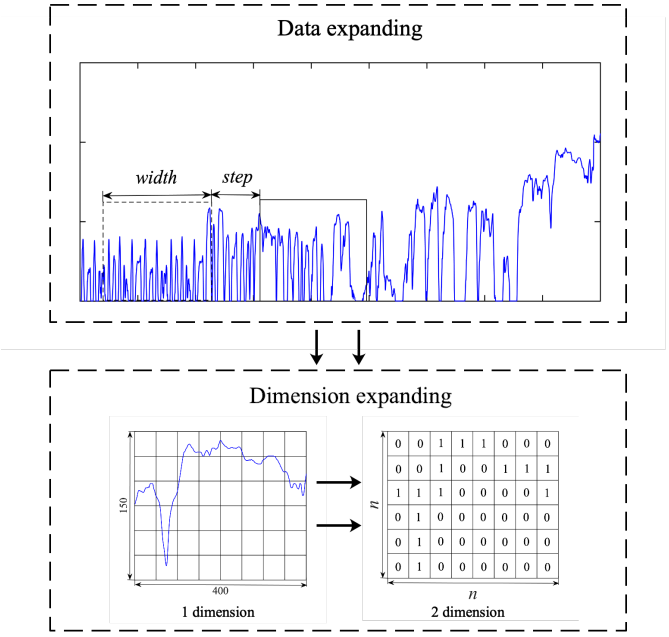


FIGURE 5: Data transforming process

the first and second convolution layers, respectively. The parameters listed in Table 2 are hyper parameters (i.e. choices for the algorithm that we set rather than learn). We use the hyperopt [26], a Python library, for optimizing the hyper parameters, to select the best model set automatically.

D. RESULTS ANALYSIS

We equally divide the dataset into training and test datasets to train and test the CNN model separately. Generally, the larger the dataset, the stronger the generalization of the model will be. Although our dataset is relatively small, we have successfully avoided overfitting by adding the KPCA layer to the architecture and achieved state-of-the-art results. The accuracy reaches 100% on the training set and 97.40% on the test set, which outperforms the other models based on different machine learning methods. The simulation was implemented at TensorFlow, running on a laptop with Intel Core i5 @ 2.3GHz and 8GB RAM. The training and testing classification results are illustrated in FIGURE6, where only a small number of samples in class 2 (UDDS) and class 3 (WVUSUB) are misidentified on the testing set.

Knowing how the classifier performs on individual classes is important as it helps to refine the system design. A receiver operating characteristic (ROC), or simply ROC curve is plotted from the confusion matrix [27] to assess the performance of the classifier on the individual classes. By computing the area under the ROC curve denoted by AUC, the quality of the classifier is comprehensively evaluated. Fig. 7 shows the ROC curve of the CNN+KPCA model on the testing set, where the AUC of classes 1 and 4 are both 1, which means

TABLE 2: The Hyper Parameters of Typical Benchmarks

Models	Parameters	Values	Parameters	Values
KNN	k	20		
	$width$	60	$step$	60
MNN	$input$	12	$hidden$	24
	$output$	4		
	$width$	80	$step$	80
KPCAMNN	$input$	5	$hidden$	9
	$output$	4	E	0.99
	$width$	80	$step$	80
CNN	n	40	m	5
	e	2	h	28
	k^1	8	k^2	16
	α	0.001	ϵ	1e-8
	β_1	0.9	β_2	0.999
	l	4		
	$width$	40	$step$	5
CNN+PCA	n	40	m	5
	e	2	h	28
	k^1	8	k^2	16
	α	0.001	ϵ	1e-8
	β_1	0.9	β_2	0.999
	E	0.95	l	4
	$width$	40	$step$	5
CNN+KPCA	n	40	m	5
	e	2	h	28
	k^1	8	k^2	16
	α	0.001	ϵ	1e-8
	β_1	0.9	β_2	0.999
	E	0.99	l	4
	$width$	40	$step$	5

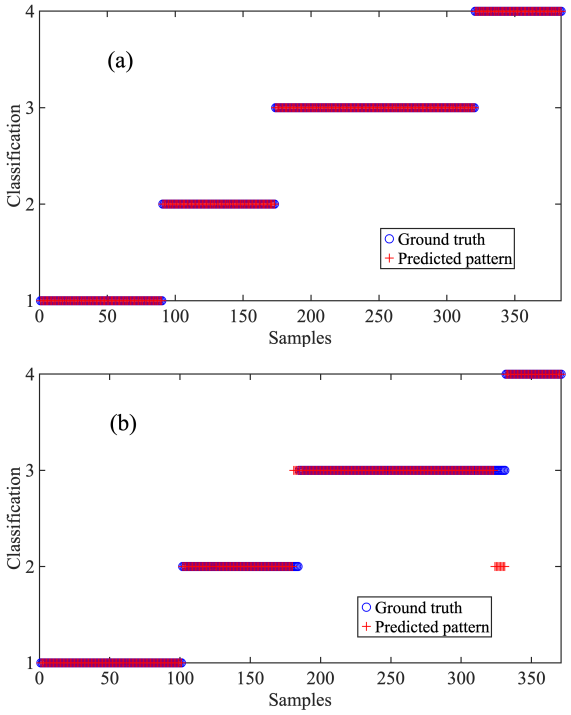


FIGURE 6: CNN+KPCA classification results (a) training (b) testing

the model has outstanding performance on those classes. The micro-average and macro-average ROC curve are calculated to evaluate the generality on the four classes.

To visualize the effectiveness of the KPCA layer, the first two components of the CNN feature extractor are selected to form the scatter plot on both training and testing sets as shown in Fig. 8. In (a) and (c), the space is formed by the first two dimensions of the originally extracted features by CNN on the training and testing sets, respectively. In (b) and (d), the space is formed by the first two components of the projected features by KPCA, where a projection of the data makes features linearly separable and this help simplify the post-fully-connected layer and improve classification accuracy.

E. COMPARISON WITH OTHER CLASSIFIERS

To evaluate the necessity and effectiveness of the KPCA strategy on the CNN feature extractor, two sets of control strategies, the standard CNN strategy and the CNN combined with PCA strategy are performed on the same dataset, respectively.

1) Typical CNN

The simplified typical LeNet-5 with only one fully-connected layer (128 hidden neurons) and dropout strategy (0.5) was selected as the classifier for real time driving pattern recognition. The structure of the standard LeNet-5 was slightly adjusted to fit in the dataset. The training and testing results on the regular dataset are illustrated in Fig. 9, where the accuracy rate are 100% and 79.78%, respectively. Although

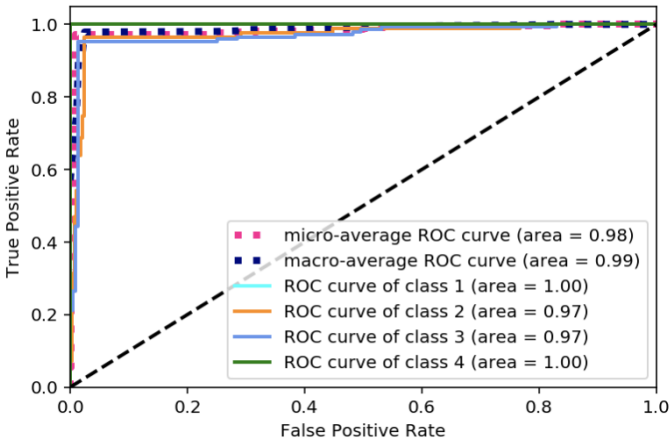


FIGURE 7: ROC curve of the CNN+KPCA model

the classifier on the training dataset achieved 100% correct rate, the testing accuracy fell far behind, which means the classifier lacked the generalization ability to handle the data point out of the training dataset, i.e. the overfitting occurred. The ROC curve on the test dataset is shown in Fig. 10, where the standard CNN classifier easily got confused between classes 2 and 3, in addition, the average accuracy rate on the test dataset is much lower than that of the proposed strategy.

2) CNN+PCA

The CNN and PCA based automated feature extractor was also evaluated for real time driving pattern recognition. The

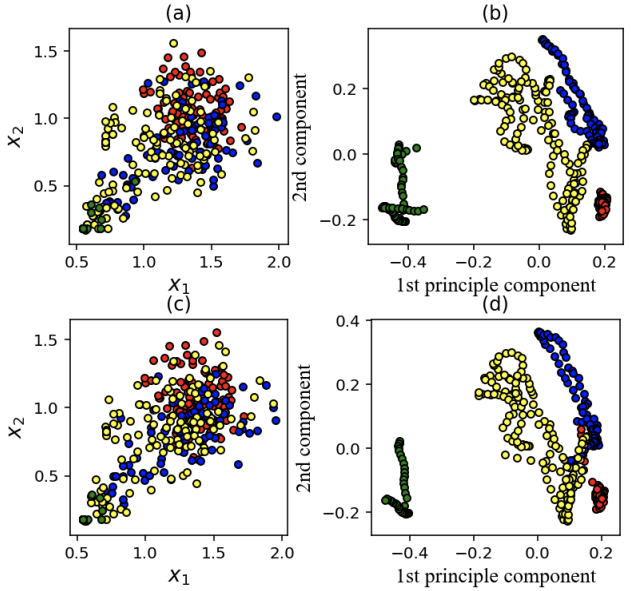


FIGURE 8: The first two components of the extracted features by CNN with or without KPCA projection. Training set: (a) Original space (b) Projection by KPCA; Testing set: (c) Original space (d) Projection by KPCA

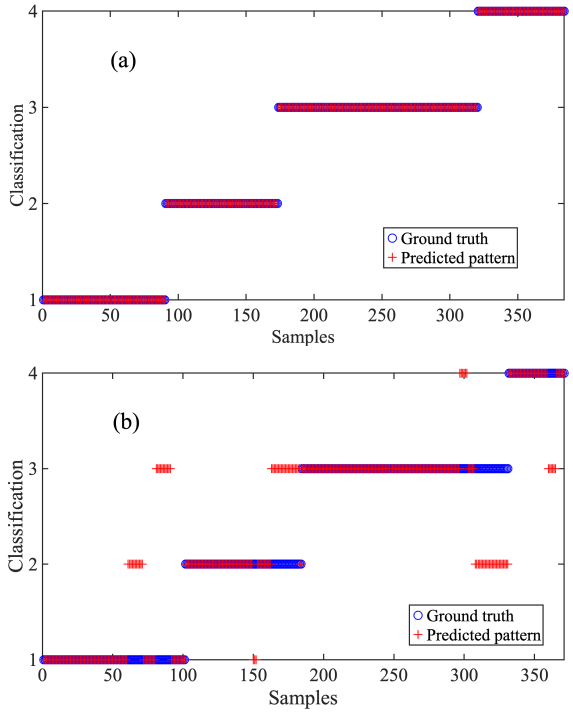


FIGURE 9: Typical CNN classification results (a) training (b) testing

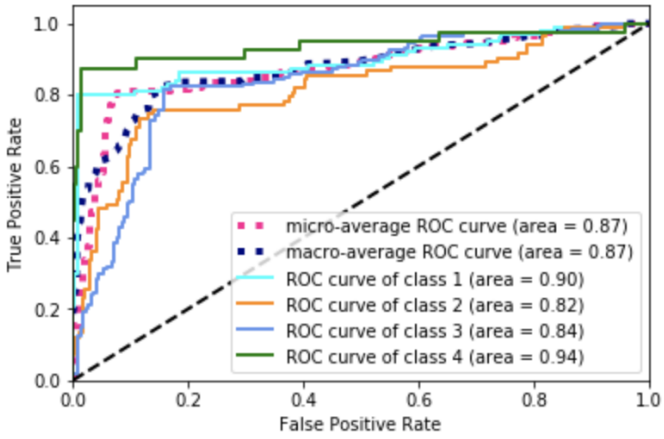


FIGURE 10: ROC curve of the typical CNN model

structure of the CNN part in the extractor was the same as the proposed extractor. In the reduction part, we apply PCA to replacing the KPCA to avoid overfitting. The classification results are illustrated in Fig. 11 and Fig. 12. The accuracy are 100% and 94.34%, respectively, where overfitting is overcame by applying the PCA strategy. The correct rate on the testing dataset is a little lower compared with the proposed strategy as KPCA is better in extracting nonlinear features.

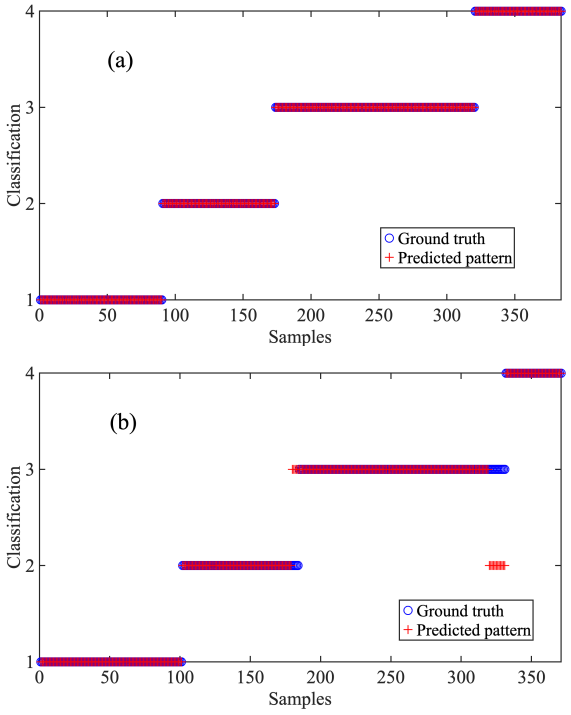


FIGURE 11: CNN+PCA classification results (a) training (b) testing

The two control strategies have proved that the standard CNN will easily suffer from overfitting when the training

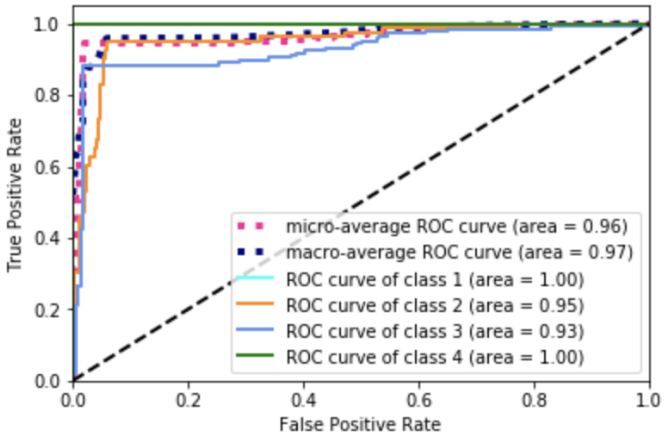


FIGURE 12: ROC curve of the CNN+PCA model

dataset is not large enough. By slightly adjusting the structure of CNN and extracting effective neurons of the fully-connected layer, overfitting would be effectively reduced. Table 3 and Fig. 13 have summarized the metrics of the three models, where there is a big gap between the training and testing accuracy in the standard CNN strategy due to the numerous parameters. In addition, the AUC of the standard CNN model is relatively lower compared with the proposed strategies, which further prove the necessity and effectiveness of the KPCA on the CNN based extractor.

TABLE 3: Metrics of the three models

Metrics	Typical CNN	CNN+PCA	CNN+KPCA
Training accuracy	1.00	1.00	1.00
Testing accuracy	0.80	0.94	0.97
AUC	0.87	0.97	0.99
Number of model parameters	1641800	3760	3472

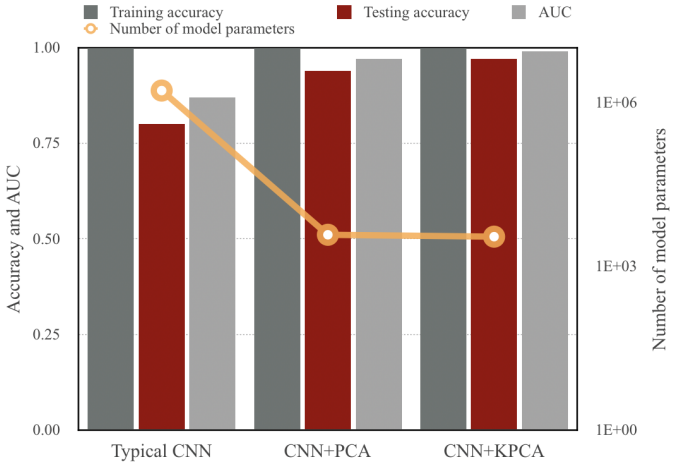


FIGURE 13: Metrics of the three Models

To prove the superiority of the proposed classifier based on automated feature extraction over the classical classifiers based on manually designed features, a variety of the traditional classifiers were trained and tested on the same dataset. Wang and Wei et al. [28], [29] have analyzed the 12 motion features that can best distinguish the driving patterns, as shown in Table 4. The following three classical classifiers were evaluated with the 12 manually designed features.

TABLE 4: Manually Designed Motion Features

Number	Motion features
1	Average speed (Km/h)
2	Maximum speed (Km/h)
3	Standard deviation (Km/h)
4	Positive average acceleration (Km/h)
5	Maximum acceleration (m/s^2)
6	Maximum deceleration (m/s^2)
7	Average deceleration (m/s^2)
8	Number of stops
9	Parking time (s)
10	Low speed time (10-25 Km/h) / total time (%)
11	Medium speed time (60-90 Km/h) / total time(%)
12	High speed time ($>$ 90 Km/h) / total time (%)

3) K-Nearest Neighbor

K-Nearest Neighbor (KNN) is one of the popularly used classifiers. This analysis generates speed feature vectors with $width = 60s$ and $step = 60s$. The features are then associated with a specific driving pattern by a KNN classifier. The classifier has the advantage that no training is required. However, the memory requirement and recognition time are demanding. On the regular test dataset, the correct rate was 81.51% with $k = 20$, while on the training dataset the correct rate was 88.02%.

The parameter k of KNN was a hyper parameter which was chosen to make the model perform best. The training and testing KNN classification results are illustrated in Fig. 14. The ROC curve of the KNN model on the testing set are given in Fig. 15, where we can find the classifier has a bad precision on classes 2 and 3.

4) Multilayer NN

Another classifier that we tested was a fully connected multilayer NN (MNN) with 3 layers. The weights of the hidden layer were obtained by training with back-propagation [30]. 12 standard features were extracted from the speed distribution with $width = 80s$ and $step = 80s$. Then these feature vectors were used to train the MNN. The accuracy rate on the regular test dataset was 85.16% with 24 hidden neurons and 84.91% with 20 hidden neurons, while on the training dataset, the accuracy rate was 91.23%. The number of the feature vectors and hidden neurons were chosen to make the model perform best. The training and testing MNN classification results are illustrated in Fig. 16. Fig. 17 shows the classifier has a bad precision on class 2. MNN based on feature extraction can achieve a relatively better result than KNN's. But the extracted features do not seem to positively contribute to the recognition task and in simplifying the classifier's structure.

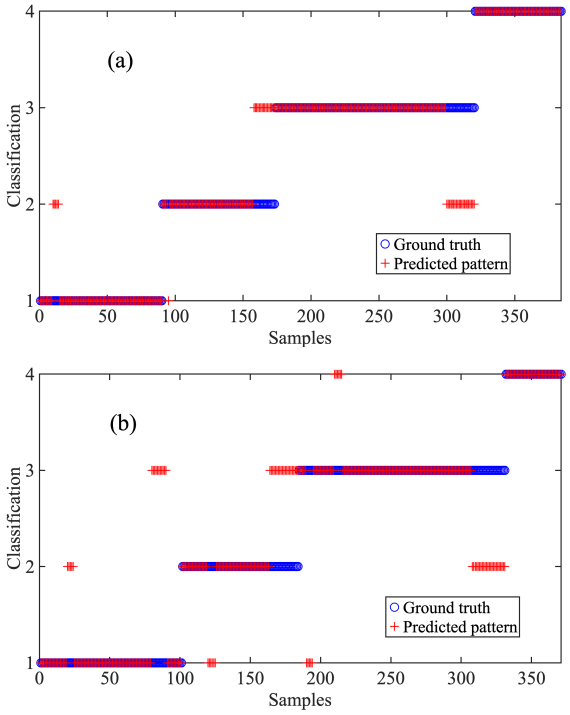


FIGURE 14: KNN classification results (a) training (b) testing

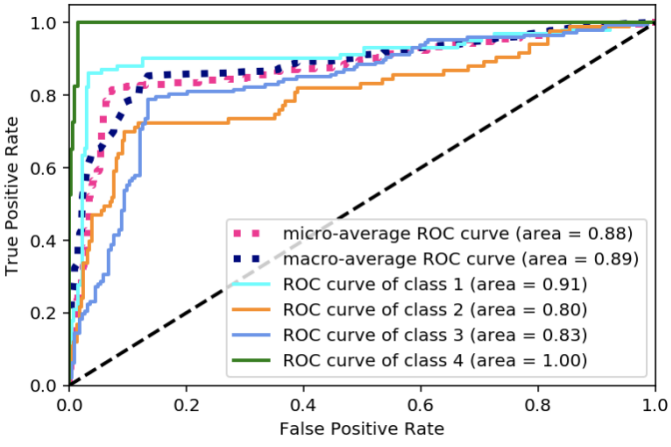


FIGURE 15: ROC curve of the KNN model

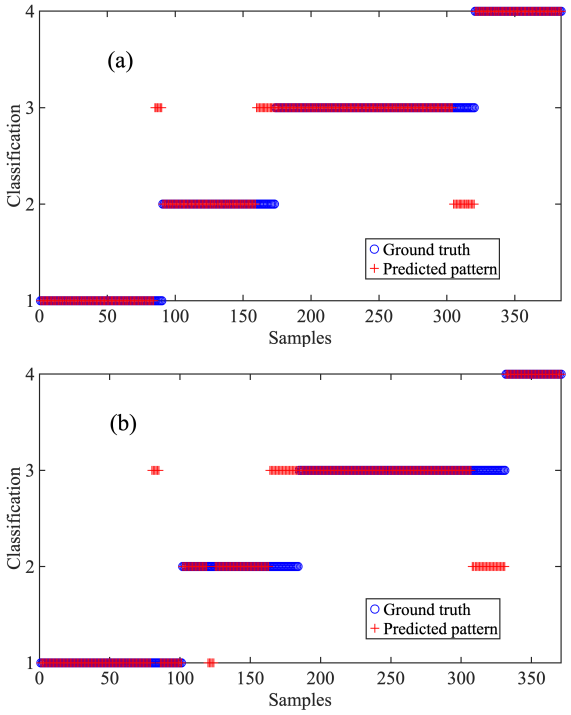


FIGURE 16: MNN classification results (a) training (b) testing

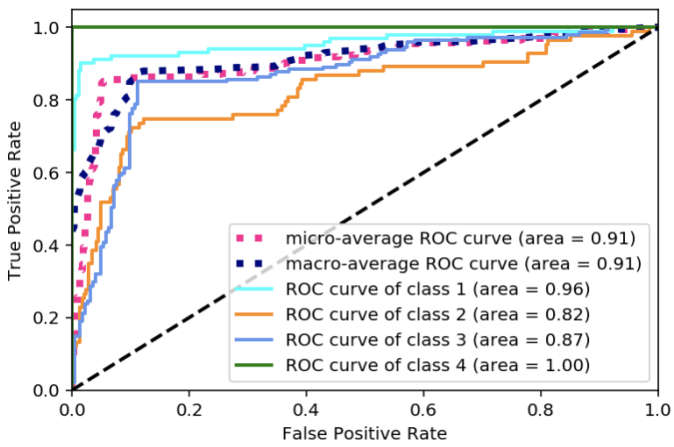


FIGURE 17: ROC curve of the MNN model

5) Kernel PCA based Multilayer NN

In [31], a preprocessing stage was constructed which computed the projection of the input pattern on the principal components as the extracted feature vectors. To compute the principal components, the mean of the input components was first computed and subtracted from the training vectors. A kernel function was then chosen to put the resulting vector into the high-dimension space. The covariance matrix of the high dimensional vectors was then computed and diagonalized by using singular value decomposition. The selected principal components represented by 5 dimensional feature vectors were used as the inputs of a multilayer classifier with 9 hidden neurons. The selected 5 principal components contained 99% information of the original data, and the number of the hidden neurons was chosen to enable the model to perform best. The accuracy on the test dataset was 91.93%, while on the training dataset the accuracy rate was 96.88%. The training and testing classification results based on kernel PCA (KPCA) based MNN (KPCAMNN) are illustrated in Fig. 18. The KPCAMNN achieves the overall better performance compared with the former two classifiers, which is shown in Fig. 19.

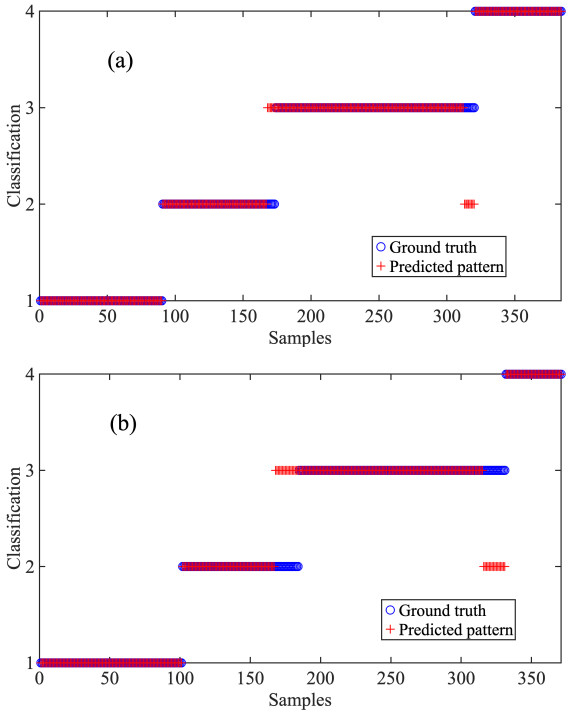


FIGURE 18: KPCAMNN classification results (a) training (b) testing

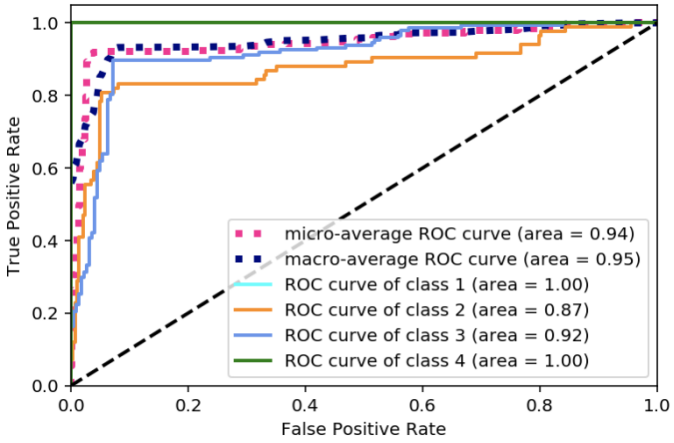


FIGURE 19: ROC curve of the KPCAMNN model

Compared with the traditional driving pattern recognition methods, the CNN+KPCA model has achieved the state-of-the-art correct rate, reaching 100% recognition accuracy on the training set and 97.40% on the testing set, whereas the best testing results from the methods based on feature extraction algorithms is 91.93%, which is a substantial leap. The model metrics including accuracy and AUC of the classifiers mentioned above are described in Table 5, from which we can conclude that the proposed framework outperforms the other methods on both accuracy and AUC.

TABLE 5: Metrics Comparison of the Classifiers

Classifiers	Training accuracy	Testing accuracy	AUC
KNN	0.8802	0.8151	0.89
MNN	0.9123	0.8516	0.91
KPCAMNN	0.9688	0.9193	0.95
CNN	1.0000	0.7978	0.87
CNN+PCA	1.0000	0.9434	0.97
CNN+KPCA	1.0000	0.9740	0.99

F. REAL DRIVING PATTERN RECOGNITION

The bus line 335 from Cicheng to Panhuo in Ningbo, China, with 41 bus stops, 11 traffic lights and 27.2Km journey crossing almost the main area of the city, was selected as the UDDS cycle to be recognized. The route map is shown in Fig. 20. The speed data were sampled from 6:00AM to 2:30PM in every 10s except the bus idle time, which is shown in Fig. 21. The classification results shown in Fig. 22 show that 95.81% samples were recognized as class 2 (i.e. UDDS cycle) with only a small part misclassified to be class 3 (WVUSUB) or class 1 (MBDC).

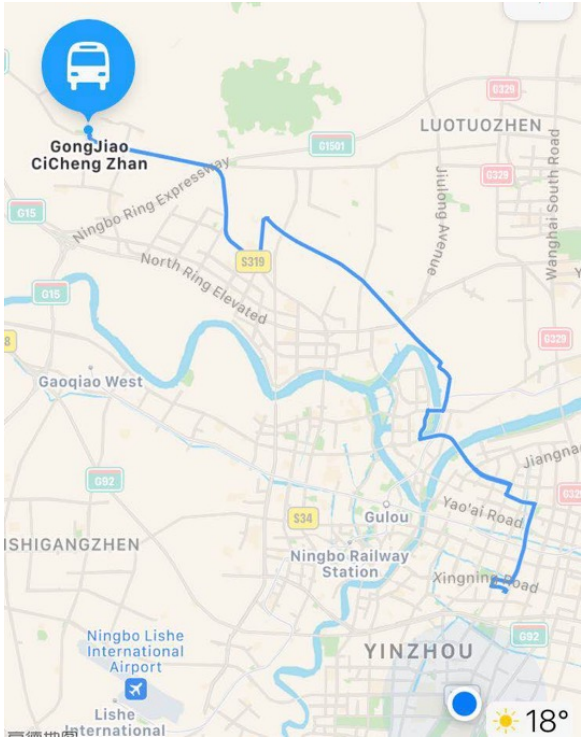


FIGURE 20: The running route of bus 335 in Ningbo

V. CONCLUSION

In this study, we have proposed a novel end-to-end approach to establish a strategy for real time driving pattern recognition applied on the speed data. Our results show that the proposed CNN+KPCA model effectively overcame the bottleneck of other traditional driving pattern classifiers based on manually extracted features. In addition, the proposed model have successfully avoid overfitting by adding the KPCA layer to

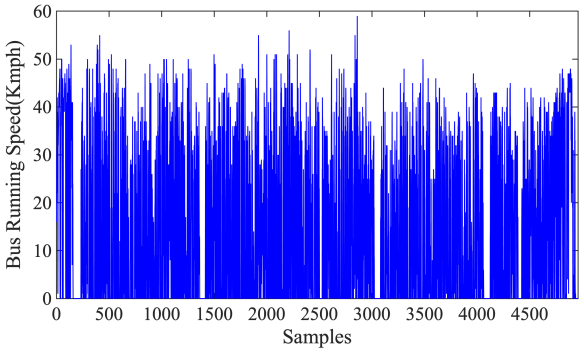


FIGURE 21: The speed samples of bus 335

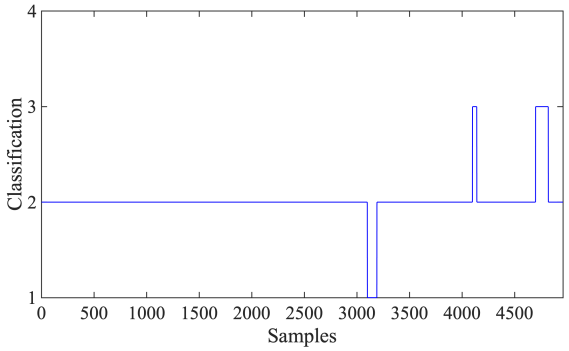


FIGURE 22: The results of real driving pattern recognition

the network architecture and the expanding dataset, which is potential when the number of the examples in the training set is not large enough. Our future goal is to apply this method to the energy management strategy of HEV to achieve efficient system control so as to save energy on vehicles.

REFERENCES

[1] Montazeri-Gh M, Ahmadi A, Asadi M, "Driving condition recognition for genetic-fuzzy HEV control, presented at the 3rd International Workshop on Genetic and Evolving Systems. IEEE, 2008.

[2] Braun A and Rid W, "Assessing driving pattern factors for the specific energy use of electric vehicles: A factor analysis approach from case study data of the Mitsubishi i–MiEV minicar," Transportation Research Part D Transport & Environment, vol. 58, pp. 225-238, 2018.

[3] Ke S et al., "Multi-mode energy management strategy for fuel cell electric vehicles based on driving pattern identification using learning vector quantization neural network algorithm," Journal of Power Sources, vol. 389, pp. 230-239, 2018.

[4] Xiaopeng T et al., "Battery state of charge estimation based on a pure hardware implementable method," DEStech Transactions on Environment, Energy and Earth Sciences, 2018.

[5] Zhang R, Tao J and Zhou H, "Fuzzy optimal energy management for fuel cell and supercapacitor systems using neural network based driving pattern recognition." IEEE Transactions on Fuzzy Systems, pp. 99, 2018.

[6] M. K. Dayeni and M. Soleymani, "Intelligent energy management of a fuel cell vehicle based on traffic condition recognition," Clean Technologies Environmental Policy, vol. 18, no. 6, pp. 1-16, 2016.

[7] A. Braun, and W. Rid, "Assessing driving pattern factors for the specific energy use of electric vehicles: A factor analysis approach from case study data of the Mitsubishi i–MiEV minicar," Transportation Research Part D Transport Environment, vol. 58, pp. 225-238, 2018.

[8] L. Si, M. Hirz and H. Brunner, "Big data-based driving pattern clustering and evaluation in combination with driving circumstances," presented at the SAE World Congress, 2018.

[9] J. D. Lee et al., "A novel driving pattern recognition and status monitoring system," presented at Pacific Rim Conference on Advances in Image and Video Technology. Springer-Verlag, 2006.

[10] H. Yu, F. Tseng and R. Mcgee, "Driving pattern identification for EV range estimation," presented at Electric Vehicle Conference, 2012.

[11] X. Zhou et al., "Fuel consumption estimates based on driving pattern recognition," presented at Green Computing and Communications. IEEE, 2013.

[12] D. Schramm et al., "Driving pattern analysis of hybrid and electric vehicles in a German conurbation including a drive system evaluation," International Journal of Advanced Mechatronic Systems, vol. 7, no. 3, pp. 158, 2017.

[13] L. Feng, W. Liu and B. Chen, "Driving pattern recognition for adaptive hybrid vehicle control," SAE International Journal of Alternative Powertrains, vol. 1, no. 1, pp. 169-179, 2012.

[14] S. Mukhopadhyay, "Deep learning and neural networks," in Advanced Data Analytics Using Python, Berkeley, CA: Apress, pp. 99-119, 2018.

[15] J. Schmidhuber, "Deep learning in neural networks," Neural Networks, vol. 61, pp. 85-117, Jan, 2015.

[16] A. Vieira and B. Ribeiro, "Deep Learning: An Overview," in Introduction to Deep Learning Business Applications for Developers, Berkeley, CA: Apress, pp. 9-35, 2018.

[17] Yudian X , Lu T , Jiang Y, "Chinese character recognition based on Gabor feature extraction and CNN," Pattern Recognition and Computer Vision, 2018.

[18] Kuo W , Ajay K, "Cross-Spectral Iris Recognition using CNN and Supervised Discrete Hashing," Pattern Recognition, 2018.

[19] Zhang B, Wang L, Zhe W, et al., "Real-Time Action Recognition with Enhanced Motion Vector CNNs," Computer Vision & Pattern Recognition, 2016.

[20] S. Haykin and B. Kosko, "Gradient-based learning applied to document recognition," in Proc. IEEE, 2009, pp. 306-351.

[21] Y. Lecun, Y. Bengio and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[22] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533-536, 1986.

[23] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," presented at International Conference on Neural Information Processing Systems, 2012.

[24] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," presented at the 8th Australasian Conference on Neural Networks, 1997.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," presented at the 3rd International Conference for Learning Representations, 2015.

[26] Bergstra J et al., "Hyperopt: a python library for model selection and hyperparameter optimization," Computational Science & Discovery, vol. 8, no. 1, 2015.

[27] Richards J.A, "Image Classification in Practice," in Proc. Springer, Berlin, Heidelberg, 2013.

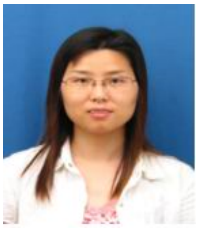
[28] Wang J et al., "Driving cycle recognition neural network algorithm based on the sliding time window for hybrid electric vehicles," International Journal of Automotive Technology, vol. 16, no. 4, pp. 685-695 2015.

[29] Wei Z , Xu Z and Halim D, "Study of HEV power management control strategy based on driving pattern recognition," Energy Procedia, vol. 88, pp. 847-853, 2016.

[30] R. Zhang and J. Tao, "Data-driven modeling using improved multi-objective optimization based neural network for coke furnace system," IEEE Transactions on Industrial Electronics, vol. 64, no. 4, pp. 3147-3155, 2017.

[31] H. He, C. Sun and X. Zhang, "A method for identification of driving patterns in hybrid electric vehicles based on a LVQ neural network," Energies, vol. 5, no. 9, pp. 3363-3380, 2012.

LIANG XIE was born in Yichun, Jiangxi, China in 1994. He received the B.E. in electrical and information engineering from Hunan University in 2017. He is currently pursuing the M.E. in control engineering from Zhejiang University, Hangzhou, China. His research interests include machine learning, deep learning and energy management in hybrid electric vehicles.



JILI TAO received the B.Sc. degree in communication engineering and M.Sc. degree in traffic information engineering and control from Central South University, Changsha, China, in 2001 and 2004, respectively, and the Ph.D. degree in control science and control engineering from Zhejiang University, Hangzhou, China, in 2007. She is currently a Professor with the Ningbo Institute of Technology, Zhejiang University, Ningbo, China. Her research interests include intelligent optimization, modeling and its applications to electronic system design and control system design.

QIANNI ZHANG received the Ph.D. degree from Queen Mary University of London, in 2007. She is currently a Senior Lecturer at Queen Mary University of London. Her research interests include Medical image analysis; 3D human modeling and animation, and various topics and applications in computer vision and pattern recognition.

HUIYU ZHOU received a Bachelor of Engineering degree in Radio Technology from Huazhong University of Science and Technology of China and a Master of Science degree in Biomedical Engineering from University of Dundee of United Kingdom, respectively. He was awarded a Doctor of Philosophy degree in Computer Vision from Heriot-Watt University, Edinburgh, United Kingdom. Dr. Zhou currently heads the Knowledge Discovery and Machine Learning Theme and is leading the Biomedical Image Processing Lab at University of Leicester. He is the Director of MSc Programmes, Coordinator of MSc Distance Learning and a Member of Research Committee at Department of Informatics. Prior to this appointment, he worked as a Lecturer (2012-17) at the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast. Dr. Zhou has published widely in the field. He was the recipient of "CVIU 2012 Most Cited Paper Award", "ICPRAM 2016 Best Paper Award in the Area of Applications" and was shortlisted for "ICPRAM 2017 Best Student Paper Award" and "MBEC 2006 Nightingale Prize". He currently serves as the Editor-in-Chief of Recent Advances in Electrical & Electronic Engineering, Associate Editor of IEEE Transactions on Human-Machine Systems, Editorial Board Member and Guest Editor of several refereed journals.