sensors

Article

A Deep Learning Framework for Driving Behavior Identification on In-Vehicle CAN-BUS Sensor Data

Jun Zhang [1,2,*], ZhongCheng Wu [1,2], Fang Li [1], Chengjun Xie [3,*], Tingting Ren [1], Jie Chen [1,2] and Liu Liu [3]

1 High Magnetic Field Laboratory, and Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China; zcwu@iim.ac.cn (Z.C.W.); lif@hmfl.ac.cn (F.L.); ttren@hmfl.ac.cn (T.R.); cj2016@mail.ustc.edu.cn (J.C.)

2 University of Science and Technology of China, Hefei 230026, China

3 Institute of Intelligent Machines, and Hefei Institute of Physical Science, Chinese Academy of Sciences, Hefei 230031, China; liuliu66@mail.ustc.edu.cn

* Correspondence: zhang_jun@hmfl.ac.cn (J.Z.); cjxie@iim.ac.cn (C.X.); Tel.: +86-139-6678-6519 (J.Z.)

Received: 23 December 2018; Accepted: 13 March 2019; Published: 18 March 2019

Abstract: Human driving behaviors are personalized and unique, and the automobile fingerprint of drivers could be helpful to automatically identify different driving behaviors and further be applied in fields such as auto-theft systems. Current research suggests that in-vehicle Controller Area Network-BUS (CAN-BUS) data can be used as an effective representation of driving behavior for recognizing different drivers. However, it is difficult to capture complex temporal features of driving behaviors in traditional methods. This paper proposes an end-to-end deep learning framework by fusing convolutional neural networks and recurrent neural networks with an attention mechanism, which is more suitable for time series CAN-BUS sensor data. The proposed method can automatically learn features of driving behaviors and model temporal features without professional knowledge in features modeling. Moreover, the method can capture salient structure features of high-dimensional sensor data and explore the correlations among multi-sensor data for rich feature representations of driving behaviors. Experimental results show that the proposed framework performs well in the real world driving behavior identification task, outperforming the state-of-the-art methods.

Keywords: driving behavior identification; deep learning; attention mechanism; CNN; LSTM; GRU

1. Introduction

Everyone has unique driving habits such as fixed speed, acceleration and braking habits, which could be considered as a fingerprint [1]. Thus, drivers' characteristics under driving conditions could be extracted through the analysis of driving behaviors. Considering different sources of data, we classify most current driving behavior identification models into three classes, that is, visual image or video-based, simulation data-based [2–4] and CAN-BUS(Controller Area Network-BUS)/smartphone multi-sensors data-based [5]. Among these, the visual data can be viewed as a special case of "multi-sensors data", and the third one, which is more effective and favorable, is our focus in this paper. Specifically, we neglect analyzing visual data due to the poor amount of training data.

Generally, multi-sensors data are made up of in-vehicle's CAN data and Smartphone data. The in-vehicle's CAN data include the steering wheel, vehicle speed, engine speed, brake position, etc., while the smartphone data include speed, orientation, three-axis accelerometer, etc. Several works proposed driver identification methods based on in-vehicle's CAN-BUS data [1,6–8]. In [9,10], deep sparse autoencoder (DSAE) was developed to extract hidden features for visualization of driving behavior, which was helpful to recognize distinctive driving behavior patterns in continuous data.

Some researchers adopted the three-axis accelerometer of an Android-based smart phone to record and analyze various driver behaviors, external road conditions [11], the degree of aggressiveness of each driver [12], and accident detection [13]. From the above works, it was concluded that driving pattern analysis is an efficient method for driver identification.

With the rapid development of Internet of Vehicles (IoV) technology and the popularization of smart terminal devices like car onboard diagnostic (OBD) devices, multi-dimensional CAN-BUS data can be easily captured for driving behavior recognition and vehicle owner identification. Driving behavior identification is essentially a classification task based on in-vehicle's CAN-BUS data. It is important to choose key features from these driving data, and find the combination of features. For instance, driver A likes to accelerate quickly at startup while driver B is used to driving at a slow speed. However, previous works developed complex feature selection techniques to improve the performance of driving behavior identification. Among them, there exist several difficulties in manual feature combination. The first one is feature explosion difficulty, which is hard for experts to explore exhaustively, especially when the number of raw features is huge. The second one is that features are difficult to design, where part of the available training data has been desensitized due to individual privacy protection, leading to impossibility in simply performing feature engineering based on common sense. Third, combined features are difficult to identify and recognize, since generally most feature interactions are hidden behind numerous data and difficult to mine, which can only be captured automatically by machine learning. Fourth, the temporal dynamics of feature activations is difficult to model explicitly. Moreover, the issue of how to effectively train the model is also a challenge, since CAN-BUS data sometimes are massive and high-dimensional, therefore large feature space will lead to a growth of parameter number, increasing the complexity of model training.

Generally, the activity recognition or identification of drivers relies on the combinations of different CAN-BUS sensor data. However, traditional feature extraction methods for driving behavior identification adopt sliding window for static and periodic activities modeling [14]. In driving behavior identification, complex high level behaviors (e.g., trip-long, day-long or more) are usually scaled up since engineering features are not related to "units of driving behaviors" but to the results of complex sequences of motor movements. However, in CAN-BUS data, multiple sensors yield multivariate time series, for instance, a single 3-axis accelerometer produces a 3-dimensional time series. Thus, it is desirable to consider the spatial dependency among multiple sensors or across the axes of accelerometers and gyroscopes, as well as the dependency along the temporal dimension.

In this paper, we propose a deep learning framework by fusing deep convolutional and recurrent neural network, denoted as attention-based DeepConvGRU and DeepConvLSTM respectively, for driving behavior identification. The outline of our approach is illustrated in Figure 1. First, in-vehicle CAN-BUS sensor data are collected for each driver to characterize the drivers' driving behaviors. Second, the time series CAN-BUS data are split into fragments by normalization and sliding window. Finally, the annotated data are fed into our proposed deep learning framework for driver behavior identification.

The main contributions are summarized as follows:

Our framework can perform automatic activity recognition on real-time multi-dimensional in-vehicle CAN-BUS sensor data, capturing local dependency among the data in temporal dimension as well as across spatial locations.

By introducing the attention mechanism, our model can capture salient structures of high-dimensional sensor data and explore the correlations among multi-channel sensor data for rich feature representations, improving the learning performance of the model.

Our framework can perform end-to-end training without any feature selection and work directly on the raw sensor data with simple pre-processing, making it universally applicable.

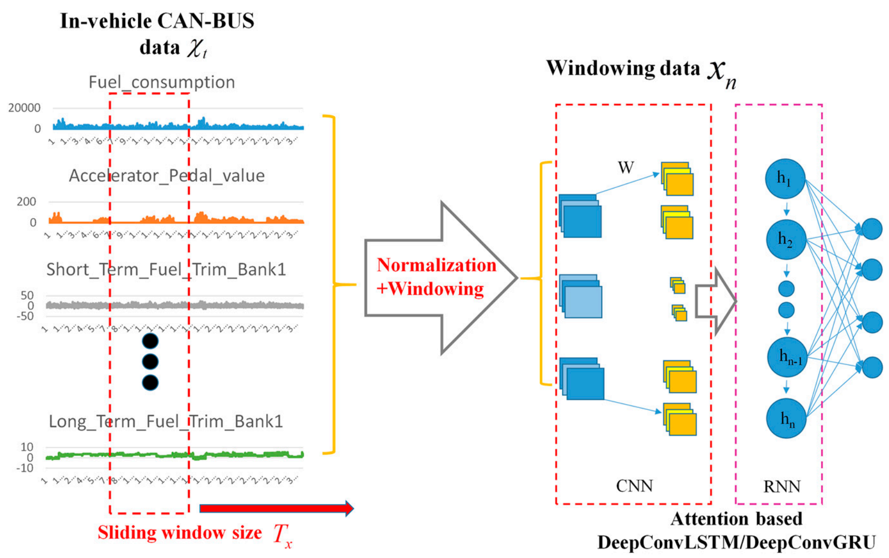

In-vehicle CAN-BUS data χ_t
Fuel_consumption
20000
0
Accelerator_Pedal_value
200
0
Short_Term_Fuel_Trim_Bank1
50
0
-50
Long_Term_Fuel_Trim_Bank1
10
0
-10
Sliding window size T_x
Normalization +Windowing
Windowing data χ_n
W
h_1
h_2
h_{n-1}
h_n
CNN
RNN
Attention based DeepConvLSTM/DeepConvGRU

Figure 1. The outline of our approach.

2. Materials and Methods

2.1. Related Works

Many state-of-the-art models were used in modeling individual driving behaviors, such as Gaussian Mixture Model (GMM) [2,6,15,16], Hidden Markov Model (HMM) [4,6,17], K-means [8], Support Vector Machine (SVM), Random Forest, Naive Bayes (NB), K-Nearest Neighbor (KNN) [1,8], Multilayer Perceptron (MLP), Fuzzy-Neural-Network (FNN), statistical method [3], Decision Tree (DT) and Symbolic Aggregate Approximation (SAX). However, most of them had various shortcomings. HMM was limited to contextual information representation, based on the hypothesis that the output observations were strictly independent and the current state was only related to the previous state (first-order Markov model). In addition, KNN was affected by unbalanced training data, which resulted in higher time complexity when calculating the distance from the unknown sample to all known samples. Moreover, the model of NB was based on the hypothesis that sample attributes were independent from each other. Therefore, NB might yield a lower classification performance when the number of sample attributes or the correlation between attributes became larger, which required enough samples to calculate the overall distribution of each class and the probability distribution of each sample. For the DT model, it had to scan and sort the data set repeatedly during model construction, which would increase the complexity and reduce the classification accuracy.

Deep learning has a great advantage in feature learning. For example, Convolutional Neural Network (CNN) [18] is mainly used for data with dense feature learning such as images and speech, while RNN and Long Short-Term Memory (LSTM) are popular choices in text homogenization and serialization of high-dimensional sparse features [19]. Driving behavior recognition involves classifying time series data captured from inertial sensors such as 3-axis accelerometers or gyroscopes. Recently, CNN has established itself as a powerful technique for activity recognition, where convolution and pooling operations were applied along the temporal dimension of sensor signals [20]. Furthermore, in most of the state-of-the-art works on CNN for activity recognition, 1D/2D convolution was employed in individual time series to capture local dependency along the temporal dimension of sensor signals [21,22]. The combination of CNN and LSTM had already offered state-of-the-art results in speech recognition, wearable activity recognition, online defect recognition of CO_2 welding, etc., where modeling temporal information was required [14,23–26]. This kind of architecture was able to capture time dependencies on features extracted by convolution operations. In this work, we focused on extracting key features using an end-to-end deep learning approach without the requirement of feature selection. In addition, features characterizing both driving behaviors and automotive running were used to represent a driver's personality.

2.2. Problem Formulation

2.2.1. In-Vehicle CAN-BUS Sensor Data Preparation and Analysis

Our models are evaluated on Ocslab driving dataset [27,28]. The dataset is used for the AI/ML based driver classification challenge track in the 2018 Information Security R&D dataset challenge held in South Korea [29]. The dataset holds a total of 94,401 records, which are created from an experiment where ten drivers labeled from "A" to "J" completed two round trips in a similar time zone from 8 p.m. to 11 p.m. on weekdays. The On Board Diagnostics 2 (OBD-II) and CarbigsPare are used as OBD-II scanner for data collection at 1 Hz sampling rate.

Originally, there are 51-Dimensional (51D) features in the dataset and the data structure of Ocslab driving dataset is depicted in Figure 2.

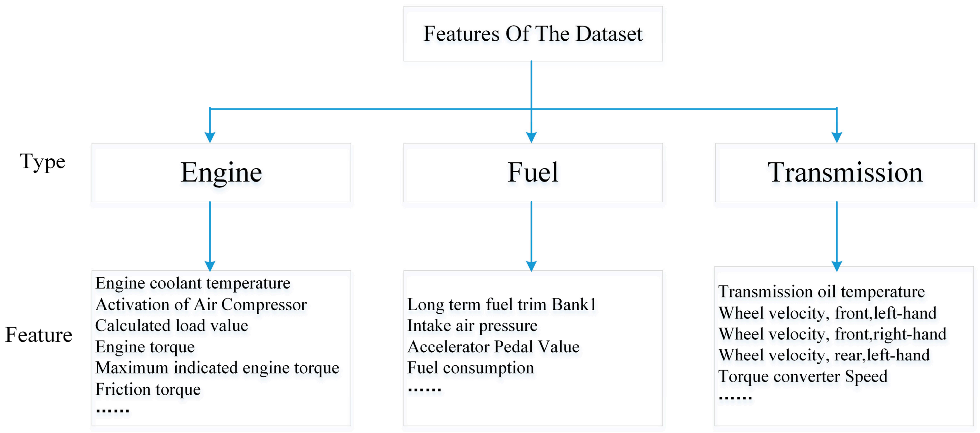

Features Of The Dataset
Type
Engine
Fuel
Transmission
Feature
Engine coolant temperature
Activation of Air Compressor
Calculated load value
Engine torque
Maximum indicated engine torque
Friction torque
......
Long term fuel trim Bank1
Intake air pressure
Accelerator Pedal Value
Fuel consumption
......
Transmission oil temperature
Wheel velocity, front,left-hand
Wheel velocity, front,right-hand
Wheel velocity, rear,left-hand
Torque converter Speed
......

Figure 2. The data structure of Ocslab driving dataset.

Some features are visualized in driver's driving pattern. Figure 3 shows the difference of revolutions per minute (RPM) when drivers B and C drove the car in the experiment.

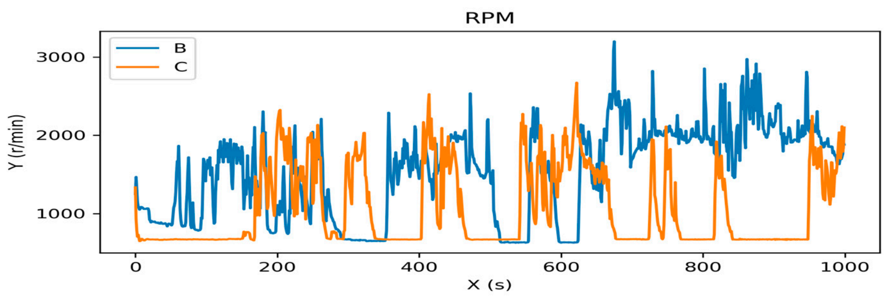


Figure 3. The visualization of revolutions per minute (RPM) of drivers B and C.

2.2.2. Data Processing

This work used all 51 original features in the dataset without complex feature selection. Before feeding to our classification model, the data was normalized and processed using sliding window technique. The input data is defined as $\chi \in \mathbb{R}^{N_\chi \times M_\chi}$. The data at time step t is defined as:

$$\chi_t = \left(\chi_t^1, \chi_t^2, \ldots, \chi_t^{N_x - 1}, \chi_t^{N_x}\right)^T \tag{1}$$

where N_χ denotes the dimensionality of χ_t, and M_χ represents the amount of dataset χ, that is, the total number of χ in all time steps.

Since the scales of features in the dataset are different, they are needed to be normalized in a classification algorithm. Specifically, the normalization process for unifying data scales is defined as:

$$\begin{aligned}\overline{\chi}_t^n &= \frac{(\chi_t^n - mean(\chi^n))}{std(\chi^n)} \\ \overline{\chi}_t &= \left(\overline{\chi}_t^1, \overline{\chi}_t^2, \ldots, \overline{\chi}_t^{N_x-1}, \overline{\chi}_t^{N_x}\right)^T\end{aligned} \tag{2}$$

where $mean(\chi^n)$ and $std(\chi^n)$ represent the mean and standard deviation of the n^{th} dimension of dataset χ, respectively.

Driving behavior is a continuous process, so sliding window technique is adopted to divide the entire data set into multiple discrete data segments by time period. In order to extract contextual features and ensure the continuity of data segments, presuming T_x is window size, data segments are extracted by the sliding window method with overlapping window. For the dataset with N_χ dimensions, the windowed sample x_i holds $D_x = T_x \times N_x$ dimensions, which are generated by

$$x_i = \left(\overline{\chi}_{t-T_\chi+1}, \overline{\chi}_{t-T_\chi+2}, \ldots, \overline{\chi}_t\right)^T \in \mathbb{R}^{N_x}, (t = T_\chi, T_\chi + \Delta t, T_\chi + 2\Delta t, \ldots) \tag{3}$$

As shown in Figure 4, the windowing dataset $X \in \mathbb{R}^{N_x \times M_x}$ is generated when x_i moves at the time axis by the time step Δt, where $N_x = N_\chi$ and M_x is the amount of the windowing dataset X.

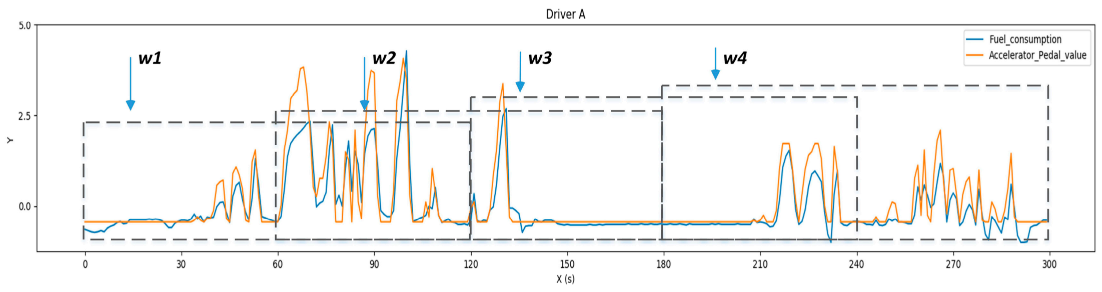


Figure 4. Overlapping sliding window method. Four windows (w_1, w_2, w_3, w_4) were obtained from the 300 samples when setting the window size T_x to 120 samples and time step Δt to 60 samples.

2.3. *Our Proposed Framework*

2.3.1. Main Procedure of Our Proposed Architecture

Compared to the structure of DeepConvLSTM proposed in [14,23–26], we introduce an attention mechanism in [30], and redesign the convolutional and recurrent layer referring to [31,32]. As shown in Figure 5, the proposed model for driving behavior identification using in-vehicle CAN-BUS sensor data consists of an input layer, middle layers and a classifier layer. D_x is the dimension of input data sample in input layer and N_y is the output categories in output layer.

The middle layers consist of convolutional layers, pooling layers, recurrent layers and a fully connected layer.

Figure 5 shows the flowchart of our model. First, a window series extracted from the CAN-BUS sensor data is passed into convolutional layers. Next, attention-based recurrent layers are used for time series feature extraction, whose inputs are the feature maps of the last convolutional layer. Lastly, the output layer, followed by the recurrent layers, is used to yield class probability distribution for driving behavior identification.

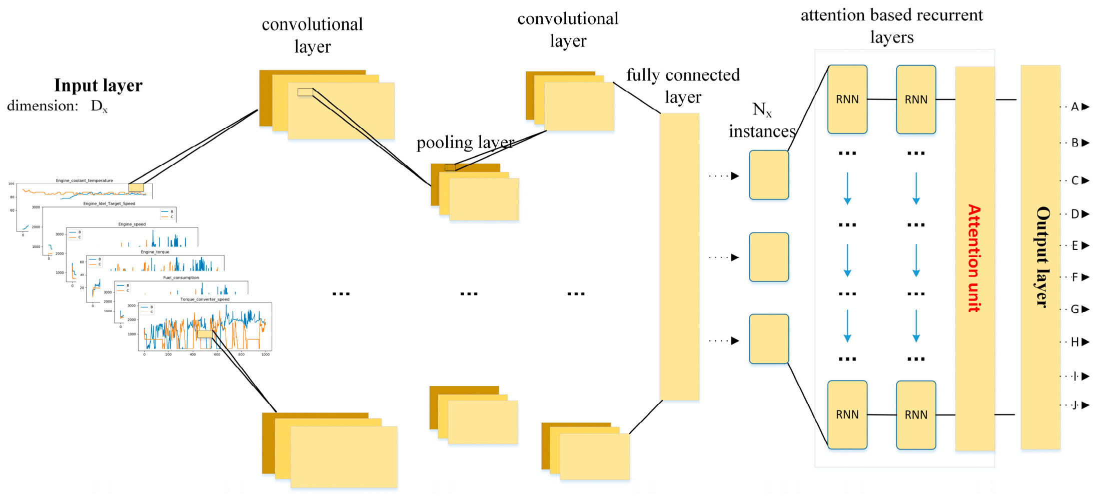


Figure 5. The main procedure of our proposed model.

2.3.2. Convolutional and Pooling Layers for Feature Extraction

Our model contains depth-wise separable convolutional layers [33] and a pooling layer in the beginning, which take convolutional operations on the input time series data. Each group of outputs of a convolutional layer is called feature map, which are regarded as features extracted from input signals. It is supposed that the number of feature map from the $(l-1)^{th}$ convolutional layer is n_{l-1}, and the size of each feature map is $m_{l-1} = w_{l-1} \times h_{l-1}$. The total number of neurons in the $l-1^{th}$ layer is $n_{l-1} \times m_{l-1}$. The k^{th} feature map output from the l^{th} convolutional layer is:

$$X^{(l,k)} = \sigma\left(\sum_{p=1}^{n_{l-1}} W^{(l,k,p)} \otimes X^{(l-1,p)} + b^{(l,k)}\right) \tag{4}$$

where σ is the ReLU activation function, $W^{(l,k,p)} \in \mathbb{R}^{u \times v}$, which is the 2D filter mapping from the p^{th} feature map of the $l-1^{th}$ layer to the k^{th} feature map of the l^{th} layer. In addition, $X^{(l,k)} \in \mathbb{R}^{w_l \times h_l}$, $w_l = w_{l-1} - w_f + 1$, $h_l = h_{l-1} - h_f + 1$, where w_f and h_f are the width and height of the filter, respectively.

Generally, the convolutional layers are followed by pooling operations, which could greatly reduce the dimension of feature maps and avoid over-fitting. The output of the pooling layer is as follow:

$$X^{(l+1)} = down\left(X^{(l)}\right) \tag{5}$$

where $down(X^l)$ is down-sampling function for the l^{th} convolutional layer $X^{(l)}$, which generally takes the maximum (Maximum Pooling) or average (Average Pooling) of all neurons in pooling region.

From equation (4) and equation (5), we can see that the first convolutional layer operates sensor data with D_x dimensions into $c_{f_1} \times m_{f_1}$ feature maps by applying 2D filters with shape $\left[h_{f_1}, w_{f_1}, c_{f_1}, m_{f_1}\right]$, where $h_{f_1}, w_{f_1}, c_{f_1}, m_{f_1}$ are respectively the filter height, filter width, input channel and channel multiplier of the 1st convolutional layer. The following pooling layer uses a kernel with shape $\left[1, h_{k_1}, w_{k_1}, 1\right]$ to down-sample feature maps, where h_{k_1}, w_{k_1} are respectively the 1st pooling layer kernel height and width.

The window inputs are split into N_x instances in time dimension. This N_x instances data is then fed into recurrent layers, in which each layer owns N_h hidden nodes.

2.3.3. Attention Based Recurrent Layer

There are two extended Recurrent Neural Network (RNN): Long Short-Term Memory (LSTM) [34] and Gated Recurrent Unit (GRU) [35]. They all use purpose-built memory cells to store information, which is helpful to find and exploit long range dependencies in time series data and thus can be further leading to more efficient driving pattern recognition. Thus, LSTM and GRU are adopted as the recurrent components that make use of the concept of gating, a mechanism based on the component-wise multiplication of inputs, which defines the behavior of each individual memory cell and decides whether to retain the state of the last moment or not, as well as to receive external inputs at this moment. LSTM is done with forget gates and input gates while GRU adopts update gates.

Time series sensor data contains more complex temporal information. Not all feature maps have the equal contribution in the identification of driving behaviors. With an attention mechanism, encoding the full input sequences into a fixed-length vector is no longer required. Thus the attention mechanism (see Figure 6) introduced by [30] is extended to capture salient structures of data, extracting more valuable feature maps than others for classification. The attention unit can also be viewed as a weighted average of output over time, where the weights could be learned automatically through context.

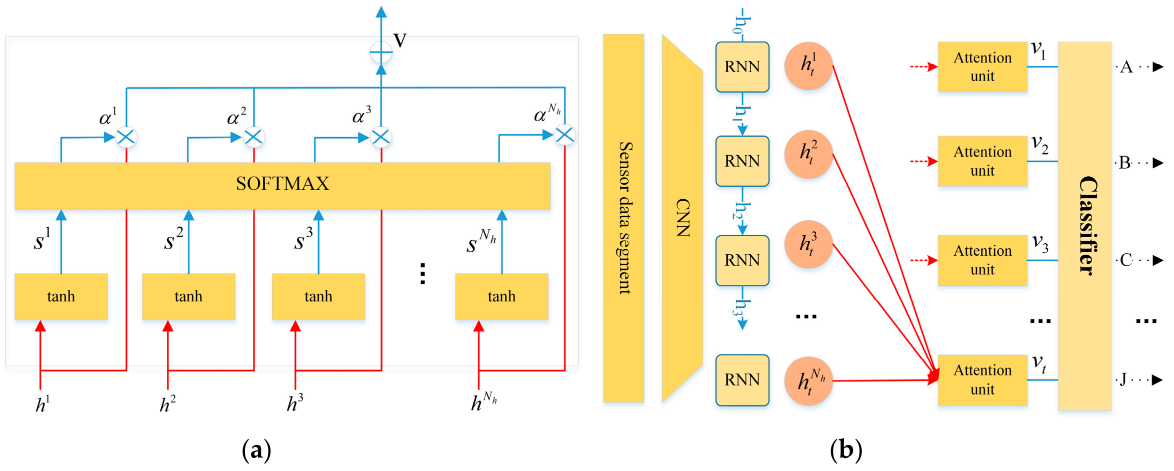


Figure 6. The attention unit.

As depicted in Figure 6a, the attention unit takes input vector $\{h^1, \ldots, h^{N_h}\}$, which is the hidden state of the recurrent layer, and outputs a contextual attention-based vector v, which is a weighted arithmetic mean of the input vector where the weights are learned based on the importance of each element of the vector. As depicted in Figure 6b, the output of the attention model v_t, which remains the importance of the representation of feature maps, is used as the input vector for the following classifier.

For each segment feature x^i at t^{th} time step, the context information is calculated by:

$$\begin{cases} s_t^i = W_s \tanh(W_h h_t^i + b_s) \\ \alpha_t^i = \dfrac{\exp(s_t^i)}{\sum_{i=1}^{N_x} \exp(s_t^i)}, \sum_{i=1}^{N_x} \alpha_t^i = 1 \end{cases} \tag{6}$$

where W_s, W_h and b_s are parameters to be learned, α_t^i is the attention weight at t^{th} time step describing the importance of the input vector. Given the current hidden state h_t of the decoder, it returns un-normalized score s_t^i. Once the scores S_t for all the nodes $\{h^1, \ldots, h^{N_h}\}$ are computed, the RNN is able to obtain α_t^i at t^{th} time step. The contextual attention-based output is:

$$v_t = \sum_{i=1}^{N_h} \alpha_t^i h_t^i \tag{7}$$

where v_t represents context vector which is a dynamic representation of the feature map at t^{th} time step.

Next, v_t is augmented to the basic LSTM and the basis formulation of LSTM [34] is below:

$$\begin{cases} i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\ c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ h_t = o_t \tanh(c_t) \\ v_t = \sum \alpha_t h_t \end{cases} \tag{8}$$

where σ is logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate, and cell input activation vectors, which are the same size as the hidden vector h and could be updated at every time step t. W_{hi} is the weight matrix of hidden-input gate and W_{xo} is the matrix of input–output gate.

Similarly, v_t is added into GRU referred to [35] and the outputs are calculated by:

$$\begin{cases} z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t = \sigma(W_t x_t + U_t h_{t-1} + b_r) \\ \overline{h}_t = \tanh(W x_t + U(r_t \circ h_{t-1})) \\ h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \overline{h}_t \\ v_t = \sum \alpha_t h_t \end{cases} \tag{9}$$

where $\circ$ is an element-wise multiplication, z_t, r_t, $\overline{h}_t$ and h_t are the update gate, reset gate, candidate activation and output activation, respectively.

2.3.4. Classifier Layer for Driving Behavior Identification

Then the output of recurrent layer $X_r = \left\{x^1, \ldots, x^{N_h}\right\}$ is fed into a classifier layer to generate the prediction $\hat{y}$. In the classifier layer, a learnable matrix W_o with a bias term b_o are used to decode X_r into $\hat{y}$, such that $\hat{y} = W_o X_r + b_o$. Therefore, the classifier layer is a fully connected layer with sharing parameter W_o and b_o.

2.4. Model Training

(A) Learning:

Since our model is a multi-class classification model, the most commonly used objective function is cross-entropy cost function, which is similar to the K-L divergence between two distributions:

$$J(w) = -\sum_{i=1}^{m}\left(y^{(i)} \log\left(\hat{y}\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}\left(x^{(i)}\right)\right)\right) \tag{10}$$

where $\left(x^{(i)}, y^{(i)}\right)$ represents the input sample with label i, and $\hat{y}\left(x^{(i)}\right)$ is the prediction of the instance $x^{(i)}$.

(B) Overfitting:

An overfitting model performs poorly since it overreacts to the given training data. Therefore, dropout is adopted to DeepConvLSTM/DeepConvGRU framework.

2.5. Model Evaluation

In order to compare our models with the state-of-the-art methods, three evaluation metrics are selected to evaluate our experiments: Accuracy, AUC [36] and weighted F1 score. Previous related work used the weighted F1 score as the primary performance metric [14]. The weighted F1 score is defined as:

$$F1 = \sum_i 2 * \omega_i \frac{precision_i recall_i}{precision_i + recall_i} \tag{11}$$

where i is class index and $\omega_i = n_i/N$ is the proportion of samples of the class i, with n_i being the number of samples of the i^{th} class and N being the total number of samples.

3. Results

Our model is evaluated and compared with other two methods, which are variants of our model created by removing the attention units from our model. It is also compared with some other state-of-the-art models [27], which are described below:

DeepConvGRU–Attention: This model has two depth wise separable convolutional layers and a pooling layer in CNN module, followed by stacked GRU with two attention-based layers.

DeepConvLSTM–Attention: Compared to DeepConvGRU–Attention, this model replaces GRU with LSTM in the recurrent layers.

DeepConvGRU: This model is similar to DeepConvGRU–Attention without attention units in model training.

DeepConvLSTM: Similarly, this model removes attention units from DeepConvLSTM–Attention.

CNN: This model owns two depth wise separable convolutional layers and a pooling layer with a softmax classifier in the output. The baseline algorithm is used to verify the effectiveness of the recurrent layers in finding and exploiting long range dependencies in time series data, which is suitable for driving pattern recognition.

LSTM: This model has two stacked LSTM layers, referred to in [37,38].

DNN: This model has two stacked hidden layers, referred to in [38].

Our model used all original 51D features to identify driving behaviors. To show the power of our end-to-end framework, feature selection referred to in [27] was implemented, selecting 15-Dimensional (15D) features from the original 51D features and deriving three statistical features for original features. In total, statistical 45-Dimensional (45D) features were obtained. Table 1 shows the selected original features and statistical features. We chose the KNN, Decision Tree and Random Forest algorithms as the baselines [27,38] as they have been proven to yield good performance.

Table 1. Selected original features and statistical features.

Selected 15D Features	Statistical 45D Features
'Long term fuel trim bank', 'Intake air pressure', 'Accelerator pedal value', 'Fuel consumption', 'Friction torque', 'Maximum indicated engine torque', 'Engine torque', 'Calculated load value', 'Activation of air compressor', 'Engine coolant temperature', 'Transmission oil temperature', 'Wheel velocity front left-hand', 'Wheel velocity front right-hand', 'Wheel velocity rear left-hand', 'Torque converter speed'	Mean Median Standard deviation

The Ocslab driving dataset was split into a training set and a test set with a ratio of 7:3 for validating the model performance. To achieve the best performance for each model in the dataset, the parameters of models were fully tuned. The hyper-parameters of compared deep models are listed in Table 2, which shows the structure of layers, window size, dropout, activation function and optimizer.

The window size T_x and Δt were set referred to [39]. As shown in Table 3, for different models, different features were chosen to get the best performance.

Table 2. Hyper-parameters of models on Ocslab driving dataset.

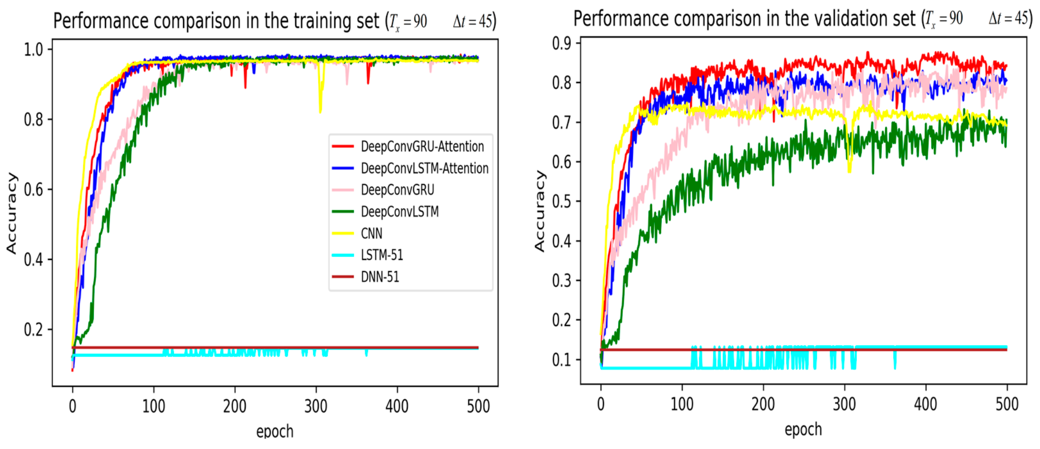
Model	Layers (*l*) [1]	Dropout	Activation Function	Optimizer
DeepConvGRU–Attention	$(1 \times 60) - 1 \times 20 - 1 \times 6 - 128 - 128 - 10$	0.5	ReLU	Adam
DeepConvLSTM–Attention	$(1 \times 60) - 1 \times 20 - 1 \times 6 - 128 - 128 - 10$	0.5	ReLU	Adam
DeepConvGRU	$(1 \times 60) - 1 \times 20 - 1 \times 6 - 128 - 128$	0.5	ReLU	Adam
DeepConvLSTM	$(1 \times 60) - 1 \times 20 - 1 \times 6 - 128 - 128$	0.5	ReLU	Adam
CNN	$(1 \times 60) - 1 \times 20 - 1 \times 6$	0.5	ReLU	Adam
LSTM	$128 - 128$	0.5	ReLU	Adam
DNN	$1000 - 1000$	0.5	ReLU	Adam

[1] '(1*60)' represents the kernel size of input-to-state convolutional layer. '1*20' and '1*6' represent the corresponding kernel sizes of state-to-state convolutional layer and pooling layer. '128' refers to the number of hidden states in the recurrent layers while '10' represents the size of attention vector. '1000' refers to the number of hidden states in the hidden layers of DNN.

Table 3. Hyper-parameters of models on Ocslab driving dataset.

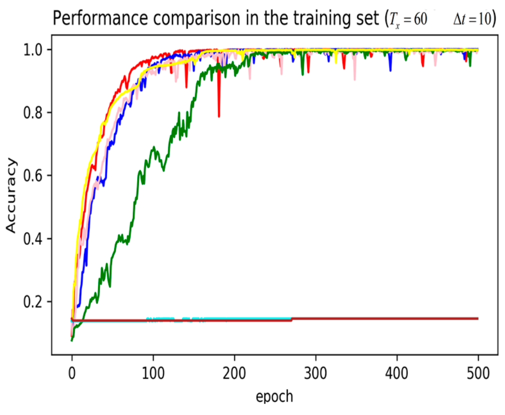
Model	Window Size (T_x)	Time Step (Δt)	Selected Features
DeepConvGRU-Attention	90/60/60	45/10/6	original 51D features
DeepConvLSTM-Attention	90/60/60	45/10/6	original 51D features
DeepConvGRU	90/60/60	45/10/6	original 51D features
DeepConvLSTM	90/60/60	45/10/6	original 51D features
CNN	90/60/60	45/10/6	original 51D features
LSTM-51	90/60/60	45/10/6	original 51D features
LSTM-15	90/60/60	45/10/6	selected 15D features
DNN-51	90/60/60	45/10/6	original 51D features
DNN-15	90/60/60	45/10/6	selected 15D features
DNN-45	90/60/60	45/10/6	statistical 45D features
KNN	90/60/60	45/10/6	statistical 45D features
Decision Tree	90/60/60	45/10/6	statistical 45D features
Random Forest	90/60/60	45/10/6	statistical 45D features

In training stage, the effects of the attention and RNN units were investigated in terms of model learning efficiency and generalization ability under Adam optimizer. The 5-fold cross-validation was used to make sure the proposed model was generalized over the dataset, in which the total data samples were divided into five parts, where four of them were used for the training model and the remaining one was employed for validation. Figures 7–12 illustrate the evaluations of the first fold training and the verification stage with respect to accuracy.

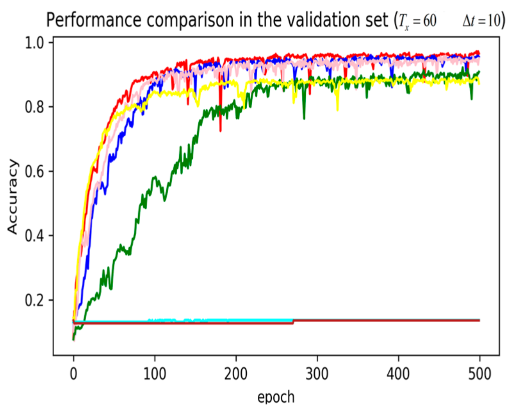


(**a**) (**b**)

Figure 7. Performance comparison of different models in accuracy using 51 original features when setting T_x to 90 samples and Δt to 45 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage.

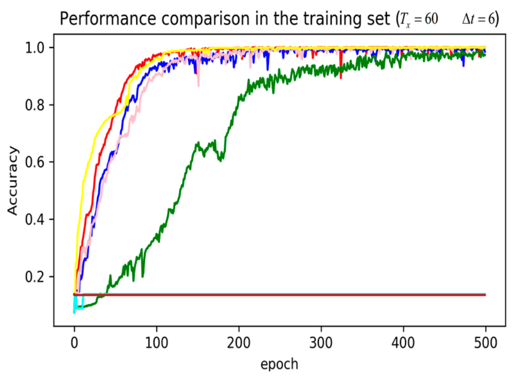


(**a**)

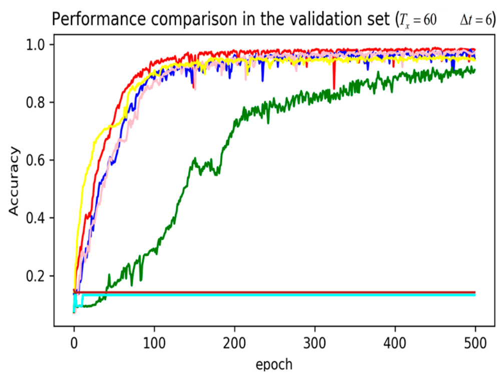


(**b**)

Figure 8. Performance comparison of different models in accuracy using 51 original features when setting T_x to 60 samples and Δt to 10 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage.

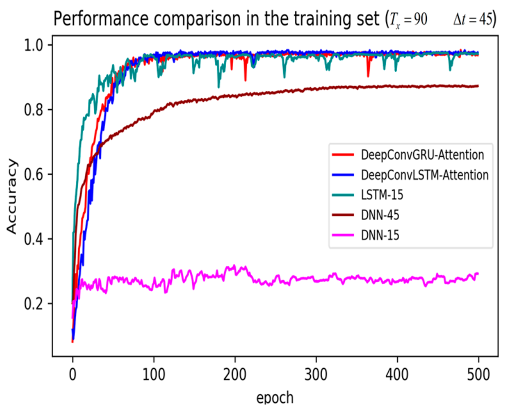


(**a**)

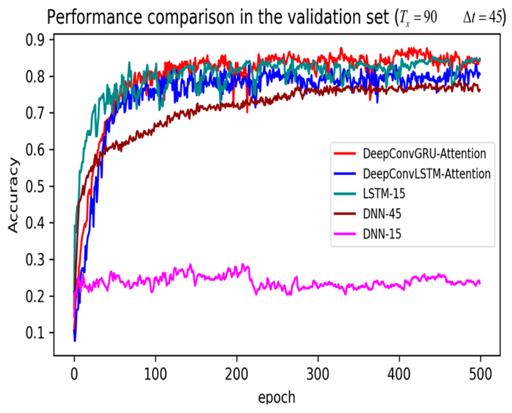


(**b**)

Figure 9. Performance comparison of different models in accuracy using 51 original features when setting T_x to 60 samples and Δt to 6 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage.

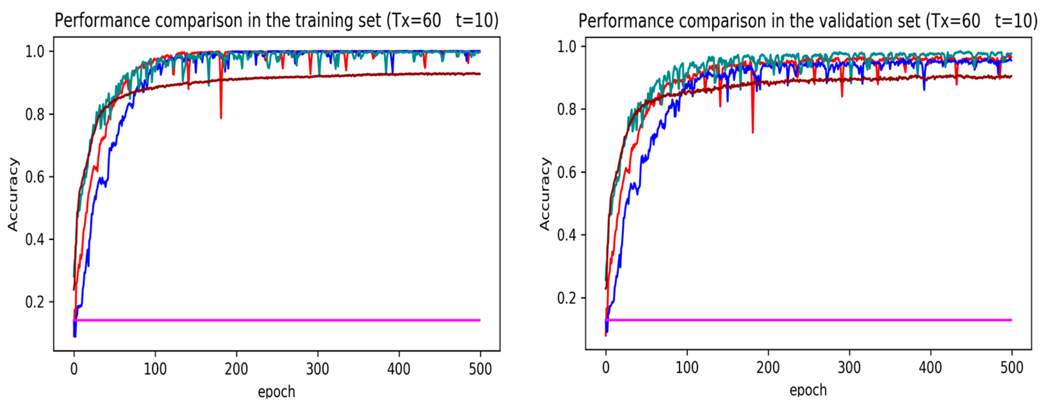


(**a**)



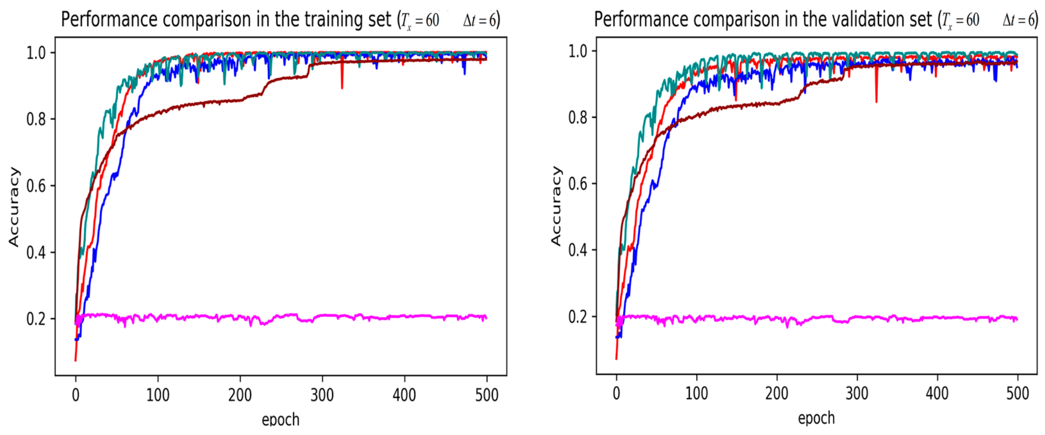
(**b**)

Figure 10. Performance comparison of different models in accuracy using 51 original features when setting T_x to 90 samples and Δt to 45 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage. Here, DeepConvGRU and DeepConvLSTM using the original 51D features performed similar to LSTM and DNN using the selected 15D features and statistical 45D features. In addition, DNN using selected 15D features also could not be converged.



(**a**) (**b**)

Figure 11. Performance comparison of different models in accuracy using original 51D features when setting T_x to 60 samples and Δt to 10 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage. Here, the attention-based DeepConvGRU and DeepConvLSTM performed better than DNN using statistical 45D features.



(**a**) (**b**)

Figure 12. Performance comparison of different models in accuracy using original 51D features when setting T_x to 60 samples and Δt to 6 samples. (**a**) Performance comparison in the training stage. (**b**) Performance comparison in the validation stage. Here, the attention-based DeepConvGRU and DeepConvLSTM also yielded good performance.

The legends of Figures 7–9 are identical. From Figure 7a, LSTM and DNN could not be converged if using all the original features without performing any feature selection. Other models could automatically select features because of the convolutional layers. From Figure 7b, DeepConvGRU and DeepConvLSTM gained better generalization ability, capturing local dependency among the temporal dimension compared with CNN. DeepConvGRU yielded faster learning efficiency than DeepConvLSTM because GRU has less parameters and therefore was easier to be converged. In Figures 8 and 9, it can be seen that the attention based DeepConvGRU and DeepConvLSTM also performed the best compared with other models. The attention mechanism made the model easier to be converged.

From the results in Figures 7–9, we can see that the attention based DeepConvGRU/DeepConvLSTM consistently outperformed the baselines. It can be noticed that DeepConvGRU made a striking performance improvement. This may be because that LSTM has more parameters than GRU, which makes it more difficult to be converged on a small dataset. The fact that DeepConvGRU/DeepConvLSTM obtained better performance than CNN may be due to the ability of RNN cells to capture temporal dynamics within the data sequences. However, the baseline CNN was only capable of modelling time sequences up to the length of the kernels. Moreover, LSTM and DNN could not be converged if using all original features. So LSTM and DNN with selected 15D features and statistical 45D features were investigated and compared with other models in Figures 10–12.

The legends of Figures 10–12 are identical. From Figures 10–12, it can be seen that the attention based DeepConvGRU and DeepConvLSTM using original 51D features without any feature selection gained similar good performance to LSTM and DNN using artificially designed features. The baseline DNN using statistical 45D features yielded poor learning efficiency and generalization ability when setting T_x to 60 samples and Δt to 10 samples. Furthermore, the baseline DNN using selected 15D features could not be converged in all cases.

To fully show the performance comparison of the models, F1 scores of the models were explored except for the models that could not be converged. The results are shown in Table 4.

Table 4. Driving behavior identification for different methods when setting T_x to 60 samples and Δt to 10 samples.

Model	T_x = 60, Δt = 10					
	Accuracy		**AUC**		**F1 Score**	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
KNN	0.812	0	0.8986	0	0.8157	0
DecisionTree	0.7432	0.0966	0.8638	0.0502	0.7402	0.1002
RandomForest	0.7049	0.0589	0.8541	0.0275	0.7565	0.0485
DeepConvGRU-Attention	0.9701	0.0052	0.9959	0.0007	0.9702	0.0052
DeepConvLSTM-Attention	0.9524	0.0138	0.9946	0.0015	0.9526	0.0137
DeepConvGRU	0.9565	0.0091	0.9959	0.0011	0.9563	0.0093
DeepConvLSTM	0.905	0.0062	0.9887	0.0008	0.9047	0.0064
CNN	0.9081	0.0096	0.993	0.0013	0.908	0.0096
LSTM-15	0.9774	0.012	0.9982	0.0008	0.9771	0.0124
DNN-45	0.9188	0.0301	0.9731	0.0164	0.9145	0.0337

Tables 4 and 5 illustrates the performance comparison of the proposed four variants of our framework compared with traditional models including CNN, LSTM, KNN, Decision Tree and Random Forest under different T_x and Δt. Experimental results showed that our framework outperformed traditional methods without any feature selection. Without feature selection, our framework also performed better than DNN and gained similar good performance to LSTM using artificially designed features. Moreover, the attention-based DeepConvGRU and DeepConvLSTM–Attention yielded better improvements than DeepConvGRU and DeepConvLSTM, respectively. In conclusion, the attention mechanism effectively helps to learn more discriminative features in time series data.

Table 5. Driving behavior identification for different methods when setting T_x to 60 samples and Δt to 6 samples.

Model	T_x = 60, Δt = 6					
	Accuracy		**AUC**		**F1 Score**	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
KNN	0.9033	0	0.947	0	0.9045	0
DecisionTree	0.8543	0.0062	0.9213	0.004	0.8537	0.0069
RandomForest	0.8739	0.0044	0.9359	0.0025	0.8934	0.0034
DeepConvGRU-Attention	0.9836	0.0015	0.9978	0.001	0.9836	0.0015
DeepConvLSTM-Attention	0.9786	0.0068	0.9978	0.0006	0.9787	0.0068
DeepConvGRU	0.9772	0.0062	0.9968	0.0008	0.9772	0.0062
DeepConvLSTM	0.9519	0.0186	0.9944	0.0013	0.9497	0.019
CNN	0.9568	0.0072	0.9984	0.0002	0.9567	0.0073
LSTM-15	0.993	0.0015	0.9996	0.0001	0.9929	0.0015
DNN-45	0.9395	0.0358	0.9682	0.0281	0.9315	0.0493

4. Discussion

From the performance comparison of our attention based DeepConvGRU/DeepConvLSTM with the baseline models without RNN unit and attention unit in the dense layer, several main findings were obtained.

First, DeepConvGRU/DeepConvLSTM reaches a higher F1 score. It is significantly more suitable for identifying disambiguate closely-related activities, which tend to differ with ordering time series data, and it is applicable for the activities that are longer than the observation window. The experimental results show that our framework can capture local dependency among the temporal dimension as well as across spatial locations.

Second, the attention mechanism makes DeepConvGRU/DeepConvLSTM gaining better generalization ability, which could automatically learn the weights of features and extract important features for the driving behavior identification.

Third, our framework outperforms traditional methods without any feature selection. Since CAN-BUS data sometimes are massive and high-dimensional, our framework is very advantageous in the case of difficult feature selection.

Furthermore, since the driving activity duration is longer than the sliding window size, experimental results showed that the model can nevertheless obtain a good performance. This might be because long driving activities are made of several short characteristic patterns, allowing the model to spot and classify the driving activity even without a complete view of the activity.

5. Conclusions

This paper presented a deep learning framework based on the combination of CNN and GRU/LSTM recurrent network to identify driving behaviors using in-vehicle CAN-BUS sensor data. In the framework, the GRU/LSTM cells were integrated into CNN to distinguish activities from similar driving behaviors. The attention based DeepConvGRU/DeepConvLSTM took advantage of learning temporal dynamics. Experimental results showed that our proposed method outperformed the traditional methods on the Ocslab driving dataset.

From the experimental results, it was also obvious that the proposed framework is able to learn features from original signals and fuse the learned features without any specific preprocessing. Surprisingly but reasonably, the attention-based DeepConvGRU achieved competitive F1 scores (0.984 and 0.970 respectively) while directly using 51-channel original sensor data. This provided a path to address a similar issue that sensor data from different sources must be automatically processed.

In the future, further researches can be conducted in the following aspects:

First, a multi-scale approach should be developed to achieve accurate activity recognition on in-vehicle CAN-BUS sensor data.

Second, due to the individual privacy protection of some driving datasets, most datasets do not disclose the complete time series data of driving behaviors from different drivers. Therefore, our framework can only be verified on a public driving behavior dataset. In the future, we need to investigate our model on more practical large-scale Naturalistic Driving Studies (NDS) datasets, such as 100-CAR [40], SHRP2 NDS [41,42], etc.

Author Contributions: J.Z. conceptualized and implemented the deep frameworks, executed the experimental work, analysed the results, drafted the original manuscript and revised the manuscript. F.L. and C.X. conceptualized the deep frameworks, visualized and analysed the results, provided feedback. T.R. was in charge of data curation and investigation. J.C. and L.L. were in charge of formal analysis and validation. Z.C.W. acquired the funding, revised the manuscript and approved the final manuscript as submitted.

Funding: This research was funded by The Science and Technology Service Network (STS) Double Innovation Project of the Chinese Academy of Sciences, the construction and application of the comprehensive management service platform for urban intelligent business travel (Grant No. KFJ-STS-SCYD-017).

Acknowledgments: The authors would like to thank Byung Il Kwak and Huy Kang Kim with Korea University, and JiYoung Woo with Soonchunhyang University for their driving dataset.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Enev, M.; Takakuwa, A.; Koscher, K.; Kohno, T. Automobile Driver Fingerprinting. *Proc. Priv. Enhanc. Technol.* **2016**, *2016*, 34–50. [CrossRef]
2. Nishiwaki, Y.; Ozawa, K.; Wakita, T.; Miyajima, C.; Itou, K.; Takeda, K. Driver identification based on spectral analysis of driving behavioral signals. In *Advances for in-Vehicle and Mobile Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 25–34.
3. Wahab, A.; Quek, C.; Tan, C.K.; Takeda, K. Driving profile modeling and recognition based on soft computing approach. *IEEE Trans. Neural Netw.* **2009**, *20*, 563–582. [CrossRef]
4. Zhang, X.; Zhao, X.; Rong, J. A study of individual characteristics of driving behavior based on hidden markov model. *Sens. Transducers* **2014**, *167*, 194.
5. Kaplan, S.; Guvensan, M.A.; Yavuz, A.G.; Karalurt, Y. Driver Behavior Analysis for Safe Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 3017–3032. [CrossRef]
6. Choi, S.; Kim, J.; Kwak, D.; Angkititrakul, P.; Hansen, J.H. Analysis and classification of driver behavior using in-vehicle can-bus information. In Proceedings of the ITS World Congress 2012, Vienna, Austria, 22–26 October 2012.
7. Kedar-Dongarkar, G.; Das, M. Driver classification for optimization of energy usage in a vehicle. *Procedia Comput. Sci.* **2012**, *8*, 388–393. [CrossRef]
8. Van Ly, M.; Martin, S.; Trivedi, M.M. Driver classification and driving style recognition using inertial sensors. In Proceedings of the Intelligent Vehicles Symposium IEEE, Gold Coast City, Australia, 23–26 June 2013; pp. 1040–1045.
9. Liu, H.; Taniguchi, T.; Tanaka, Y.; Takenaka, K.; Bando, T. Visualization of Driving Behavior Based on Hidden Feature Extraction by Using Deep Learning. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2477–2489. [CrossRef]
10. Liu, H.L.; Taniguchi, T.; Takano, T.; Tanaka, Y. Visualization of driving behavior using deep sparse autoencoder. In Proceedings of the Intelligent Vehicles Symposium IEEE, Ypsilanti, MI, USA, 8–11 June 2014; pp. 1427–1434.
11. Fazeen, M.; Gozick, B.; Dantu, R.; Bhukhiya, M.; González, M.C. Safe Driving Using Mobile Phones. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1462–1468. [CrossRef]
12. Dai, J.; Teng, J.; Bai, X.; Shen, Z.; Xuan, D. Mobile phone based drunk driving detection. In Proceedings of the 4th International Conference on Pervasive Computing Technologies for Healthcare, Munich, Germany, 22–25 March 2010; pp. 1–8.
13. Zaldivar, J.; Calafate, C.T.; Cano, J.C.; Manzoni, P. Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones. In Proceedings of the IEEE Conference on Local Computer Networks, Bonn, Germany, 4–7 October 2011; pp. 813–819.
14. Ordóñez, F.J.; Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* **2016**, *16*, 115. [CrossRef] [PubMed]
15. Miyajima, C.; Nishiwaki, Y.; Ozawa, K.; Wakita, T.; Itou, K.; Takeda, K.; Itakura, F. Driver modeling based on driving behavior and its evaluation in driver identification. *Proc. IEEE* **2007**, *95*, 427–437. [CrossRef]
16. Wakita, T.; Ozawa, K.; Miyajima, C.; Igarashi, K.; Itou, K.; Takeda, K.; Itakura, F. Driver identification using driving behavior signals. In Proceedings of the IEEE Intelligent Transportation Systems Conference, Toronto, ON, Canada, 17–20 September 2006; pp. 396–401.
17. Meng, X.; Lee, K.K.; Xu, Y. Human Driving Behavior Recognition Based on Hidden Markov Models. In Proceedings of the IEEE International Conference on Robotics & Biomimetics, Sanya, China, 15–28 December 2007.
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.

19. Foland, W.; Martin, J.H. CU-NLP at SemEval-2016 Task 8: AMR Parsing using LSTM-based Recurrent Neural Networks. In Proceedings of the International Workshop on Semantic Evaluation, San Diego, CA, USA, 16–17 June 2016; pp. 1197–1201.
20. Ha, S.; Choi, S. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In Proceedings of the International Joint Conference on Neural Networks, Vancouver, BC, Canada, 24–29 July 2016.
21. Duffner, S.; Berlemont, S.; Lefebvre, G.; Garcia, C. 3D gesture classification with convolutional neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Florence, Italy, 4–9 May 2014.
22. Zeng, M.; Nguyen, L.T.; Yu, B.; Mengshoel, O.J.; Zhu, J.; Wu, P.; Zhang, J. Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors. In Proceedings of the International Conference on Mobile Computing, Applications and Services, Berlin, Germany, 12–13 November 2015.
23. Sainath, T.N.; Vinyals, O.; Senior, A.; Sak, H. Convolutional, long short-term memory, fully connected deep neural networks. In Proceedings of the IEEE International Conference on Acoustics, South Brisbane, Australia, 19–24 April 2015; pp. 4580–4584.
24. Bilgera, C.; Yamamoto, A.; Sawano, M.; Matsukura, H.; Ishida, H. Application of Convolutional Long Short-Term Memory Neural Networks to Signals Collected from a Sensor Network for Autonomous Gas Source Localization in Outdoor Environments. *Sensors* **2018**, *18*, 4484. [CrossRef] [PubMed]
25. Liu, T.; Bao, J.; Wang, J.; Zhang, Y. A Hybrid CNN–LSTM Algorithm for Online Defect Recognition of CO_2 Welding. *Sensors* **2018**, *18*, 4369. [CrossRef] [PubMed]
26. Nguyen, V.; Nguyen, M.; Choi, J.; Kim, Y. NLOS Identification in WLANs Using Deep LSTM with CNN Features. *Sensors* **2018**, *18*, 4057. [CrossRef] [PubMed]
27. Kwak, B.I.; Woo, J.; Kim, H.K.; Huy, K. Know your master: Driver profiling-based anti-theft method. In Proceedings of the 14th Annual Conference on Privacy, Security and Trust, Auckland, New Zealand, 12–14 December 2016.
28. Driving Dataset. Available online: http://ocslab.hksecurity.net/Datasets/driving-dataset (accessed on 19 November 2018).
29. AI/ML Based Driver Classification Challenge Track. Available online: http://datachallenge.kr/challenge18/vehicle/introduction/ (accessed on 22 February 2019).
30. Attention Mechanism. Available online: https://blog.heuritech.com/2016/01/20/attention-mechanism/ (accessed on 19 November 2018).
31. Li, X. cnnPlusLSTM. Available online: https://github.com/lixiaoyu0575/cnnPlusLSTM (accessed on 19 September 2018).
32. Saeed, A. Implementing a CNN for Human Activity Recognition in Tensorflow. Available online: http://aqibsaeed.github.io/2016-11-04-human-activity-recognition-cnn/ (accessed on 19 September 2018).
33. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807.
34. Graves, A. Generating sequences with recurrent neural networks. *arXiv* **2013**, arXiv:1308.0850.
35. Chung, J.; Gulcehre, C.; Cho, K.H.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
36. Fawcett, T. An introduction to ROC analysis. *Pattern Recogn. Lett.* **2005**, *27*, 861–874. [CrossRef]
37. Jiménez, D.; Hernández, S.; Fraile-Ardanuy, J.; Serrano, J.; Fernández, R.; Álvarez, F. Modelling the Effect of Driving Events on Electrical Vehicle Energy Consumption Using Inertial Sensors in Smartphones. *Energies* **2018**, *11*, 412. [CrossRef]
38. Wang, Y.; Ho, I.W. Joint Deep Neural Network Modelling and Statistical Analysis on Characterizing Driving Behaviors. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1–6.
39. Carvalho, E.; Ferreira, B.V.; Ferreira, J.; De Souza, C.; Carvalho, H.V.; Suhara, Y.; Pentland, A.S.; Pessin, G. Exploiting the use of recurrent neural networks for driver behavior profiling. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017.
40. Guo, F.; Fang, Y. Individual driver risk assessment using naturalistic driving data. *Accid. Anal. Prev.* **2013**, *61*, 3–9. [CrossRef] [PubMed]

41. Antin, J.; Lee, S.; Hankey, J.; Dingus, T. *Design of the In-Vehicle Driving Behavior and Crash Risk Study: In Support of the SHRP 2 Naturalistic Driving Study*; SHRP Report; Transportation Research Board: Washington, DC, USA, 2011.
42. Bärgman, J.; Lisovskaja, V.; Victor, T.; Flannagan, C.; Dozza, M. How does glance behavior influence crash and injury risk? A 'what-if' counterfactual simulation using crashes and near-crashes from SHRP2. *Transp. Res. F* **2015**, *35*, 152–169. [CrossRef]