Convolutional Neural Network With Adaptive Regularization to Classify Driving Styles on Smartphones

Mohammad Mahdi Bejani and Mehdi Ghatee

***Abstract*— Driving style evaluation by smartphones depends on the quality of the features extracted from sensors data. Typically, these features are extracted based on experiments, expertness, or heuristics. In more modern approaches, some automatic methods such as convolutional neural network (CNN) are used to extract features including obvious and hidden ones. We also used the CNN on acceleration data collected by smartphones to extract the knowledge regarding driving style, vehicle, environment, and human characteristics. We found that this novel idea was more successful for evaluating the driving style compared with the previous machine learning algorithms. However, we faced over-fitting in the training process of the CNN and to avoid this, we proposed the state-of-the-art learning method applying two adaptive regularization schemes called adaptive dropout and adaptive weight decay. To evaluate these techniques, first, we checked the results on three popular large-scale datasets. When we proved the efficiency, we utilized them on two transportation data sets. In transportation-modes dataset, the accuracy was at least 95.8%'; and regarding the driving-style dataset, the classification accuracy was 95%. Thus, the adaptive regularized CNN is an amazing option for driving style evaluation on smartphones.**

***Index Terms*— Deep learning, driving style evaluation, smartphone sensors, convolutional neural network, adaptive regularization, over-fitting.**

I. Introduction

SMARTPHONES have been extensively used in different transportation systems in recent years. As some instances, Eftekhari and Ghatee [1] used a state-transition engine for transportation mode detection. Dabiri and Heaslip [2] detected the transportation mode by a CNN on GPS data. Zadeh *et al.* [3] developed a warning system to protect vulnerable road users. Wahlström *et al.* [4] also surveyed smartphone-based systems for vehicle applications. Among these systems, driving style evaluation has been addressed by many researchers. The results of these systems can be used by insurance telematics market including pay-as-you-drive or usage-based insurance [5]. Also, the smartphone version of these systems can help drivers reduce the fuel consumption causing diminished omission of CO_2. Chhabra *et al.* [6] reviewed many systems for driver's style evaluation by considering different input signals collected by camera or smartphone sensors. These applications extract some useful features from smartphone sensors and classify or cluster the samples to evaluate the driving styles [7]–[9]. To this end, different algorithms have been used including fuzzy inference engine [10], Bayes classifier [11], probabilistic classifier [12], threshold-based method [13]–[15], and ensemble learning algorithm [16]. In the latter work, several features are extracted from acceleration in maneuvers to evaluate driving styles. Nevertheless, their approach is dependent on recognized maneuvers. Also, they need to access some extra-data from the car and traffic levels to enhance the accuracy. However, consideration of all effective features is difficult and such knowledge should be provided by other systems. Further, there are a large number of hidden features which cannot be directly extracted from smartphone data. In these situations, deep learning algorithms such as CNN are very useful, see e.g., [17] for image processing and speech recognition [18] problems. These modern algorithms can extract many features for classification purposes. In smartphone applications, they were also used by Dabiri and Heaslip [2] for transportation mode detection from GPS data. Also, Wishan *et al.* [19] proposed a CNN for driver characterization. Lv *et al.* [20] proposed a deep model based on the auto-encoder network for predicting the traffic flow. The input of their systems was the data collected from loop detectors. To the best of our knowledge, CNN has not been used on acceleration data to evaluate the driver's style. Thus, we apply a CNN for feature extraction from acceleration data and evaluate the driving style based on these features.

Nevertheless, the results of CNN can be corrupted because of over-fitting. Both deep shallow models have the over-fitting problem. There are many papers that have dealt with over-fitting in ITS field such as [21]–[24]. Meanwhile, CNN consists of many hyper-parameters which are initialized or should be adjusted through the training process by different techniques [25]–[28]. However, these methods are extremely time-consuming and assign fixed values for hyper-parameters throughout the training process. In numerous experiments, the hyper-parameters should be adjusted with respect to the training conditions. For such cases, we apply adaptive hyper-parameters in CNN and change them along the training process. We focus on regularization techniques to control the over-fitting in CNN and decide about the regularization parameters with respect to the magnitude of over-fitting.

Manuscript received April 18, 2018; revised September 27, 2018 and December 4, 2018; accepted January 28, 2019. Date of publication February 18, 2019; date of current version February 3, 2020. The Associate Editor for this paper was A. Che. *(Corresponding author: Mehdi Ghatee.)*

The authors are with the Department of Computer Science, Amirkabir University of Technology, Tehran 15875-4413, Iran (e-mail: mbejani@aut.ac.ir; ghatee@aut.ac.ir).

Digital Object Identifier 10.1109/TITS.2019.2896672

This concluded adaptive regularized CNN is called CNNAR and is used on smartphones data to generally evaluate the driving styles. This approach is similar to that of [29] and does not evaluate the driving style for a single maneuver. Rather, we consider a driving time-interval and evaluate the driving style throughout this interval. We consider the following kinds of features in each time interval:

- Obvious features: They can be collected directly from raw data. For example, the variance or the maximum of acceleration in a maneuver is obvious.
- Hidden features: They cannot be understood explicitly from raw data. For example, the effect of vehicle stability system on the acceleration sensed by occupants or the impact of road surface slippage on driving control is hidden and cannot be extracted directly from the smartphones' data.

Since there are numerous hidden features affecting the driving style evaluation, CNN is a good option for feature extraction. Also, in theory, a feature is important when its covariance with the other extracted features is almost zero. However, determining the covariance for hidden features is impossible. Instead, we use a CNN on the acceleration data collected by smartphones to extract hidden and obvious features simultaneously regardless of their importance.

In what follows, we describe some preliminaries in Section II. Then, we describe a new system for driving style evaluation based on CNN in Section III. Section IV addresses a new solution for over-fitting in CNN. Section V presents the results, and finally Section VI ends the paper with a brief conclusion.

II. Preliminaries

Artificial neural networks (ANN) are common machine learning algorithms with a great training ability and good generalization. They have different structures. Layered ANN is one of the most practical classifiers with numerous neurons in each layer which should be defined by the user. The power of layered ANN with three layers and enough neurons in the middle layer has been proven in the literature. However, finding the appropriate number of neurons is not possible theoretically and until now nobody has proven that three layers is the best option. Recently, deep networks have been introduced with many layers. Although they are not explainable but can solve very complex classification problems. They have a great potential to extract important features and can remove unimportant features. Indeed, they transform the raw data into a suitable internal representation or feature vector whose learning is simpler than the original data [30]. Over the last few years, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training deep neural networks [31]. One of the proper types of deep network is CNN, which is described further.

A. CNN Architecture

CNN belongs to supervised learning algorithms which extracts obvious and hidden features automatically. There are many points behind CNN to achieve the appropriate results [30]. In the current paper, we apply CNN models for classification purposes, considering acceleration data. In the training phase of this CNN, a batch of windowed acceleration samples is fed as the input. The tensor shape of the input is $B \times 1 \times M \times C$ where B, M, and C represent the number of training samples, size of the window, and the number of the axes of the accelerometer, respectively. Based on the error of the output layer, the parameters of the CNN are adjusted. There are many algorithms which minimize this error function such as Kingma and Ba [32], which is used in our work. The topologies of CNN for training and testing phases are similar, except the value of B which is considered as 1 throughout the testing process.

This CNN also includes several convolutional and pooling layers as the feature extractor and feature selector, respectively. A convolutional layer consists of a set of filters. Each filter is a window (matrix) moving with a specific step on the inputs and performing an operation on inputs to extract the necessary features. We consider different settings of filters, where for each setting and each movement, the filter covers a specific region of the input space and multiplies them by their corresponding values in the filter. The summation of these values is sent to a nonlinear function such as ReLU [33] to obtain a nonlinear model.

In each pooling layer, we define some new regions by considering 4×2 or 4×1 rectangles on the output of the previous layer. Then, we substitute the maximum or average of cells of any rectangle instead of the entire rectangle. This process reduces the dimension of the outputs.

After applying some pairs of convolutional and pooling layers, we employ several fully connected layers similar to multi-layer perceptron with ReLU activation functions in neurons. The final layer is a fully connected layer with softmax activation function. In Fig.1, two CNNs are sequentially used for the transportation mode detection and the driving style evaluation. To tune-up the initial values of the CNN, we follow the method of [34] and [35].

III. Driving Style Evaluation System

In this section, we propose a new driving style evaluation system called Ex-CADSE. This system is an extended version of CADSE which has previously been introduced by Bejani and Ghatee [16]. CADSE system consists of 4 subsystems: smartphone calibration, feature clustering, context awareness, and decision fusion. In the Ex-CADSE system, the first subsystem is extended to a preprocessing subsystem for transportation mode detection by CNNAR and smartphone calibration. The second and the third subsystems are merged into a new subsystem to classify the driving styles applying CNNAR. Finally, the last subsystem remains unchanged. The architecture of Ex-CADSE is presented in Fig. 1. In both subsystems, for classification purposes, since the data are imbalanced, the traditional CNN is not successful. Thus, we use CNNAR for mode detection and driving style classification in Ex-CADSE system. We discuss about the details of CNNAR in Subsection V-B. Before that, we further describe the functionalities of the two presented subsystems.

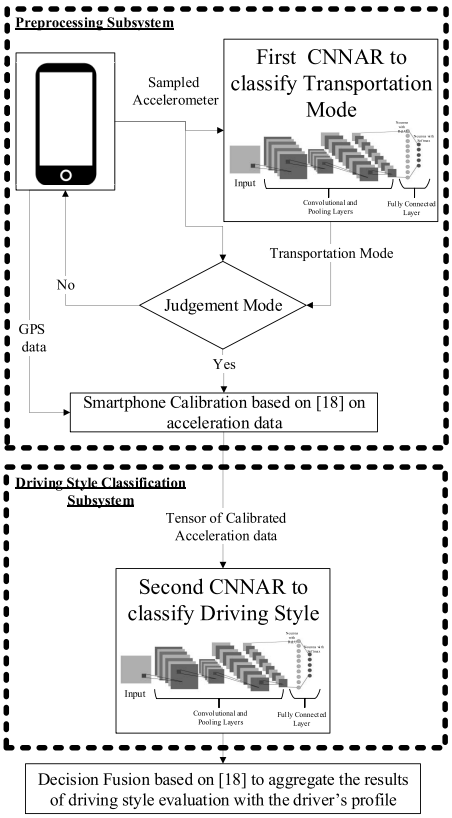

Fig. 1. A new architecture for driver evaluation system (Ex-CADSE).

A. Preprocessing Subsystem

This subsystem consists of two major modules: i) a CNNAR for classification of transportation mode; and ii) a module that performs smartphone calibration. As GPS sensor consumes large amounts of battery charge and calibration of smartphone needs the GPS data, the first CNNAR classifies the transportation mode. The CNNAR receives the raw acceleration data in three axes. These samples are saved to a tensor with $W \times 1 \times 3$ size, where W is the window length of the sampling data. CNNAR classifies the samples into five transportation modes. These results are then sent to judgement mode module. In this module, if the transportation mode is a predefined class, the acceleration data and GPS data are sent to smartphone calibration module as discussed completely in [16]. For example, when we need to evaluate a car driver, the time interval that she(he) drives is sent to the next module. The same can be done for bus drivers, heavy vehicle drivers, etc. To cover all of these situations, we use judgement mode in Ex-CADSE. To show the complexity of the transportation mode classification and to understand why CNNAR has been used for this task, we used t-SNE algorithm [36] and visualized the samples of transportation-modes dataset in the first row of Fig. 2. As one can see, in sub-figure (b), the features of the raw data cannot be classified linearly. On the other hand, sub-figure (a) demonstrates that the features, extracted from the last pooling layer of CNNAR, are almost linearly separable whereby five classes can be directly recognized. This suggests that use of CNNAR is necessary for this subsystem.

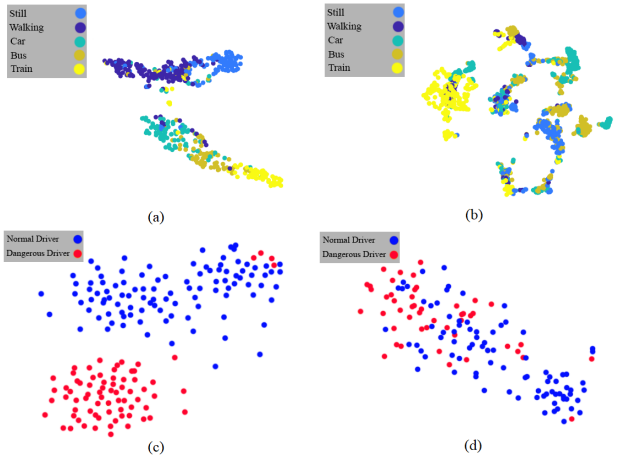

Fig. 2. The first and the second rows are the transportation-modes and the driving-styles datasets visualization by t-SNE. In sub-figures (a) and (c) the features of the last pooling layer of CNNAR are visualized while in sub-figures (b) and (d) the features of raw data are visualized.

B. Driving Style Classification Subsystem

To classify the driving style, there are two main approaches: the maneuver-based classification and the overall classification. In maneuver-based driving style evaluation [16], the quality of maneuvers is assessed. However, this kind of evaluation is limited; since there are some unknown maneuvers, while the lengths and the start time of maneuvers are not clear, and traffic conditions affect driving styles. However, in overall classification [29], the driving style is evaluated in a long time interval without focusing on several predefined maneuvers. Thus, many problems of maneuver-based classification are resolved in this approach. Another important problem in driving style evaluation is the way of feature selection. The proposed system extracts some features from the context automatically using CNNAR and then classifies the driving styles. The results of CNNAR should be sent to the decision fusion subsystem to make the final decision.

To clarify the reason of usage of CNNAR for driving style evaluation, we again visualized the samples of driving-styles dataset [16] by t-SNE algorithm [36]. The results are presented in the second row of Fig. 2. The sub-figure (c) displays the visualization results for the features extracted from the last pooling layer of CNNAR, while the sub-figure (d) illustrates the corresponding results for features of the raw data. The classes of the former sub-figure are almost separable, where dangerous (red points) and normal drivers (blue points) can be recognized easily. However, the samples of the latter sub-figure cannot be classified linearly and their patterns are critically tangled together. In these cases, the shallow models usually fail to classify the samples. Really, our experiments show that the accuracies of driver classification by SVM with linear kernel, SVM with RBF kernel, SVM with polynomial kernel, MLP, K-NN, Decision Tree and RBF on the acceleration data are 67%, 73%, 70%, 62%, 66%, 65% and 61%, respectively. When more different sensors data are collected together with the acceleration, Table VII shows the accuracies of shallow models are still less than that of the proposed deep model that considers just acceleration. Thus, it is defensible to

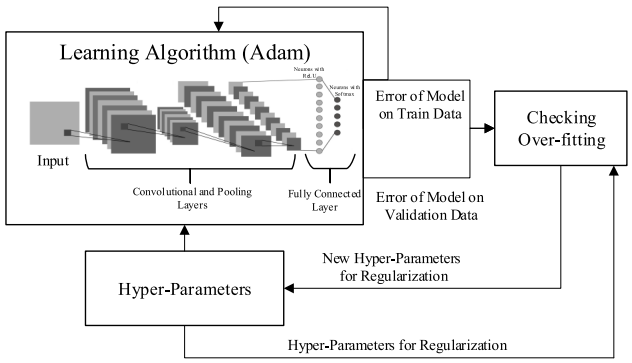

Fig. 3. The adaptive regularization framework for over-fitting controlling.

apply CNNAR to extract the useful features for classification of driving styles. Meanwhile, some points of dangerous drivers belong to the normal drivers cluster, since a dangerous driver sometimes drives like as a normal driver under some congested conditions. When CNNAR is used in the long interval time of driving, we can distinguish these situations and the decision fusion subsystem in Ex-CADSE can properly evaluate drivers. Now, to implement CNNAR on driving-styles dataset [16], since the accelerometer data have been collected at a 2Hz sampling rate, we store them in a matrix with $N \times W$ size, where N and W are the numbers of sensor axes and length of the window, respectively. This matrix is then sent to CNNAR, which classifies the samples into dangerous and normal classes. The output of the CNNAR is further sent to a module for decision fusion. The details of this module have been completely introduced in [16].

IV. Adaptive Regularized CNN

Over-fitting is the biggest drawback in classification problems by CNN. The reason is that CNN includes numerous hyper-parameters and these parameters increase with the problem complexity, where some methods should be used to control over-fitting. When the samples of different classes are imbalanced, the over-fitting corrupts the results completely. For example, since the number of dangerous drivers is usually less than that of normal drivers, CNN cannot classify the drivers correctly. Further, each person spends less time on driving compared with the other actions, and so the smartphone data for motorized mode is sparse. All these cause CNN not to be able to classify the motorized mode with great reliability. These are our reasons for extending an adaptive regularized version of CNN (CNNAR) in Ex-CADSE. Dropout and weight decay are two solutions for controlling the over-fitting, but determining their parameters requires a trial-and-error procedure. Instead, we propose a new framework called adaptive regularization whose structure is illustrated in Fig. 3. In this framework, hyper-parameters are adjusted with respect to the over-fitting conditions. In what follows, the hyper-parameters of dropout and weigh decay are adapted by applying this framework.

A. Adaptive Dropout Procedure

In this procedure, dropout layers are added following each of the fully connected layers of CNN. Every dropout layer i receives a *keeping probability* denoted with kp_i which indicates how many of synaptic weights of this layer participate in the training process. In each epoch t of training process, with respect to $kp_i(t)$ probability, some of the synaptic weights are selected randomly and are trained by the learning algorithm. Consider n_{fc} dropout layers. To implement the dropout procedure, we need to adjust hyper-parameters $(kp_i(t),\ i = 1, \ldots, n_{fc})$. Since over-fitting appears when the model contains more parameters than its ability to justify by the data [37], we define over-fitting criterion for epoch t as the following:

$$\upsilon(t) = \frac{MSE_V(t)}{MSE_{Train}(t)} \tag{1}$$

where for epoch t, $MSE_V(t)$ and $MSE_{Train}(t)$ are the mean-squared error on the validation data and on the training data, respectively. When the value of $\upsilon(t)$ is large, $MSE_{Train}(t)$ is small and (or) $MSE_V(t)$ is large. Thus, CNN memorizes the training data or cannot generalize the classification on the new data. Conversely, when $\upsilon(t)$ is small (close to 1), $MSE_{Train}(t)$ and $MSE_V(t)$ are close together. In the latter case, if $MSE_{Train}(t)$ is small, $MSE_V(t)$ is small and the training process is successful. Otherwise both of $MSE_{Train}(t)$ and $MSE_V(t)$ are large and the training process should be continued.

Now, to adjust $kp_i(t)$ $(i = 1, \ldots, n_{fc})$, we define two thresholds, $M_1(t)$, $M_2(t)$ and then we change $kp_i(t)$ with respect to $c_i(t)$. For this change, we state the following two rules:

- **Rule I:** If the over-fitting value is great, decrease $kp_i(t)$, i.e.,

$$\text{If } \upsilon(t) > M_1(t) \quad \text{then } kp_i(t) = \frac{kp_i(t-1)}{c_i(t)}$$

- **Rule II:** If the over-fitting value is small, decrease the effect of dropout with the following:

  If $\upsilon(t) < M_2(t)$ and $max\{kp_i(t) \mid i = 1, \ldots, n_{fc}\} < 1$,

  then

$$kp_i(t) = c_i(t) kp_i(t-1)$$

Now, we initialize $kp_i(0) = 1$ for all $i = 1, \ldots, n_{fc}$. Thus, we need to determine $c_i(t)$ and the thresholds $M_1(t)$ and $M_2(t)$. In the following, we state the process of finding $c_i(t)$. We postpone calculation of $M_1(t)$ and $M_2(t)$ to Subsection IV-D.

It is worthy to note that, $c_i(t)$ is related to the number of layer and the number of neurons in that layer. With respect to these numbers, we use fuzzy rules III-V.

- **Rule III:** When i is great, $c_i(t)$ is chosen as a small number.
- **Rule IV:** When the number of neurons is great, the value of $c_i(t)$ is great.
- **Rule V:** When $\upsilon(t)$ is great, $c_i(t)$ is selected as a great number.

To apply these rules, firstly, we define $\tilde{n_i}$ as:

$$\tilde{n_i} = \frac{n_i}{\max\{n_j \mid j = 1, \ldots, n_{fc}\}} \tag{2}$$

where n_i is the number of neurons in i^{th} layer. Now, define:

$$s_i(t) = \upsilon(t)\tilde{n_i}\frac{n_{fc}}{i}, \quad i = 1, \ldots, n_{fc} \tag{3}$$

Thus, the rules III-V can be expressed as the following formula for adjusting $c_i(t)$:

$$c_i(t) = \begin{cases} \beta^- & \text{if } s_i(t) \text{ is small} \\ \beta & \text{if } s_i(t) \text{ is medium} \\ \beta^+ & \text{if } s_i(t) \text{ is great} \end{cases} \tag{4}$$

where β^+, and β^- are two great and small numbers and β is chosen between these values. To adjust the appropriate values for these numbers, we applied several CNNs with dropout procedure on five big datasets and the dropout procedure with $\beta^- = 1.2$, $\beta = 1.5$, and $beta^+ = 2$ performs appropriately.

B. Adaptive Weight Decay

In weight decay, a controlling term is added to the following error function of CNN:

$$Error_{new} = Error + \alpha||w||_F^2 = MSE_{Train} + \alpha||w||_F^2, \tag{5}$$

where $||w||$ is the Frobenius norm of synaptic weights. α is also a fixed scaling parameter between the error value and decay term. When a synaptic weight $|w_{i,j}|$ converges to zero, the corresponding link between two nodes is pruned. Note that, a great α causes diminished effect of the training process while a small α declines the effect of weight decay term, whereby we do not have enough control on over-fitting. Thus, we define an adaptive weight decay $\alpha(t)$ for each epoch t. We initialize $\alpha(0)$ as a small value like 10^{-8}. In training process, $\alpha(t)$ is set by the following rules:

- **Rule VI:** If $\upsilon(t) > M_1(t)$ then $\alpha(t) = c^+(t)\alpha(t-1)$.
- **Rule VII:** If $\upsilon(t) < M_2(t)$ then $\alpha(t) = c^-(t)\alpha(t-1)$.

where $c^+(t)$ and $c^-(t)$ are defined in Eq. 6 and Eq. 7, respectively. In adaptive weight decay, when the difference between the training error and the validation error is great, the over-fitting is high and by choosing a great value for $\alpha(t)$, we can decrease the synaptic weights, which is equivalent to reduction of the nonlinearity of the model. To pursue this approach, we can do the following:

1) Experimental studies: Similar to Subsection IV-A, we compared different CNNs and obtained $c^+(t) = 2$ and $c^-(t) = 0.5$ regarding to the maximum accuracy.
2) Theoretical studies: When we try to minimize the training error, a condition on $\alpha(t)$ should be imposed such that the effect of over-fitting is not neglected. Trivially, when $\alpha(t) = \frac{MSE_{Train}(t)}{||w||_F^2}$, the priorities of training error and weight decay term are almost similar. By increasing $\alpha(t)$ the effect of weight decay increases. Thus, under over-fitting conditions, the value of $\alpha(t)$ should be chosen greater than or equal to this value. By combining the results of the experimental and theoretical studies, one can state the following:

$$c^+(t) = max\{2, \frac{MSE_{Train}(t)}{||w||_F^2\alpha(t-1)}\} \tag{6}$$

Conversely, if the over-fitting is small, $\alpha(t)$ should be chosen less than $\frac{MSE_{Train}}{||w||_F^2}$, to give an opportunity for the training process to reduce the training error. In this case, we have an upper bound on α and an experimental result which can be combined in the following formula:

$$c^-(t) = min\{0.5, \frac{MSE_{Train}(t)}{||w||_F^2\alpha(t-1)}\}. \tag{7}$$

C. Training Process

Now, it is possible to modify all of the stochastic optimization algorithms to train CNN applying both of the adaptive regularization procedures. In Algorithm 1, we state the details of a stochastic optimization scheme based on the gradient directions. Note that the previous learning algorithms for CNN have either not considered the effects of over-fitting or included the effects of over-fitting in all of the epochs. However, in the initial epochs, the classification problem is under-fitting and the extra-computation is not feasible for minimizing the training error as well as minimizing over-fitting. To resolve this major drawback, in Algorithm 1, we apply an adaptive regularization procedure just when the over-fitting is considerable. To implement this approach, we check the over-fitting conditions for all of epochs and when $\upsilon(t)$ is high, we call either adaptive dropout procedure or adaptive weight decay procedure. In this algorithm, $\vec{w}(t)$ is a vectorized shape of synaptic matrix $w(t)$. Also *DescentDirection* function depends on the learning algorithm e.g. Adam [32] or gradient descent [38]. In our implementation, *DescentDirection* function receives the gradient of the error function as the input and returns a scaled moving direction to improve the results. This function implements some stochastic changes on the gradient [38] and on the step-size to improve the results of the gradient-based optimizer. Further, *TreshCom* function receives the training error along with validation error and returns the over-fitting thresholds. The details of this function are given in the next subsection.

D. Over-Fitting Thresholds

To implement *TreshCom* function for determining the over-fitting thresholds, one can present the mathematical models (8) and (9) to find the values of $M_1(t)$ and $M_2(t)$ respectively. The first model considers a pre-defined parameter $r > 1$ and finds the minimal threshold $M_1(t)$ such that the validation error of the next epoch does not exceed r times of the training error. the closer r is to 1, the greater the sensitivity to the over-fitting will be. On the other hand, selection of $r = 1$ causes divergence. Based on the experiments on standard datasets, we obtained $r = 150$ as an appropriate factor in our implementation. The second model finds the maximum of threshold $M_2(t)$ where the validation error of the next epoch is less than the training error plus a pre-defined positive value r' (close to 1). In ours study, we obtained $r' = 5$ experimentally.

$$\begin{aligned} M_1(t) = & \textbf{ Minimize } \upsilon(t) \\ & \textbf{s.t. } r\ MSE_{Train}(t) > MSE_V(t) \end{aligned} \tag{8}$$

$$\begin{aligned} M_2(t) = & \textbf{ Maximize } \upsilon(t) \\ & \textbf{s.t. } MSE_V(t) - MSE_{Train}(t) < r'. \end{aligned} \tag{9}$$

Algorithm 1 Gradient Based Optimization Algorithm for CNN With Adaptive Weight Decay or Adaptive Dropout Procedures

Require: $\theta(0) = [w(0), b(0)]$: A matrix of initial synaptic weights $w(0)$ and bias vectors $b(0)$ of CNN.
Require: $E(\theta(0); Data)$: Error of CNN on samples belonging to $Data$.
$t = 0$ as the index of epoch.
$kp_i(0) = 1; \forall i = 1, \ldots, n_{fc}$ and $\alpha = 10e-8$
while the error of CNN is not acceptable **do**
  $t \leftarrow t + 1$.
  Set $g(t) = \bigtriangledown_\theta E(\theta(t-1); Data_{Train}) + 2\alpha\vec{w}(t)$ as the gradient of (5).
  Get the scaled moving direction

$$d(t) = DescentDirection(g(t)).$$

  $\theta(t) = \theta(t-1) - d(t)$
  $MSE_{Train}(t) = E(\theta(t); Data_{Train})$
  $MSE_V(t) = E(\theta(t); Data_{Validation})$
  $\upsilon(t) = \dfrac{MSE_{Train}(t)}{MSE_V(t)}$
  $[M_1(t), M_2(t)] = TreshCom(MSE_{Train}(t), MSE_V(t))$.
  if adaptive weight decay regularization procedure is selected **then**
    Apply (6) and (7) to determine $c^+(t)$ and $c^-(t)$.
    if $\upsilon(t) > M_1(t)$ **then**
      $\alpha(t) = c^+(t)\alpha(t-1)$
    end if
    if $\upsilon(t) < M_2(t)$ **then**
      $\alpha(t) = c^-(t)\alpha(t-1)$
    end if
  end if
  if adaptive dropout regularization procedure is selected **then**
    Apply (3) and (4) to evaluate $s_i(t)$ and $c_i(t)$.
    if $\upsilon(t) > M_1(t)$ **then**
      $kp_i(t) = \dfrac{kp_i(t-1)}{c_i(t)}$
    end if
    if $\upsilon(t) < M_2(t)$ **then**
      $kp_i(t) \leftarrow c_i(t)kp_i(t-1)$
    end if
  end if
end while
return CNN with the best parameters $\theta(t)$

Since, there is no direct relationship between the training error and validation error, we define a statistical measure to approximate the solutions of these problems. To this end, we considered different datasets. For model (8) we evaluated the constraints of all 10 epochs, and when their constraints were met, we stored the value of $\upsilon(t)$. Otherwise, we continued the experiments and finally the minimum over the obtained $\upsilon(t)$ was considered as $M_1(t)$. The same procedure was followed to obtain $M_2(t)$ from model (9). In our study, $M_1(t) = 162$ and $M_2(t) = 4$ have been obtained.

V. Results

In this section, we compare two adaptive regularization techniques including dropout and weight decay with non-adaptive regularization. All of the training processes are implemented on GPU (GTX 960 2GB). The Tensorflow (version 1.4.0) framework is used to implement CNNs. In what follows, we initially evaluate the performance of our CNNAR on three popular datasets. Then, we evaluate the performance of CNNAR in two subsystems of Ex-CADSE.

A. CNNAR on Popular Datasets

To represent the effects of adaptive regularization on the training process of CNN, we measure the accuracy, MSE, and over-fitting criterion $\upsilon(t)$. Initially, we examine three datasets including SVHN [39], STL [17], and CIFAR-10 [40]. To prove the efficiency of the proposed method, we define three phases: recognition of hyper-parameters, evaluation of different regularization schemes, and transportation applications. The first and the second phases are presented in this section, with the third phase described in the next section.

For recognizing hyper-parameters, we need to evaluate CNNAR on a standard dataset such as SVHN. This dataset is a real-world image dataset for developing machine learning and object recognition algorithms [39]. Fig.4-(a) displays the performance measures of CNN without adaptive regularization. In this training process, after each epoch, MSE of validation and MSE of test increase. Following epoch 400, the minimum value of $\upsilon(t)$ is about 150. Fig.4-(b), presents the same results using adaptive dropout. As can be observed, $kp_i(t)$ diminished when the discrepancy between the training error and the validation error grows. When $kp_i(t)$ drops, the learning algorithm provides an opportunity to find a better solution with a lower MSE. When the discrepancy between the training error and validation error decreases, $kp_i(t)$ rises thereby improving the accuracy of the model. As can be seen, this phenomenon has happened in epochs 130, 160, 190, and 220. In Fig.4-c, the performance measures of CNN with constant dropout are presented when $kp_i(t) = 0.7$ for every epoch t. This value of $kp_i(t)$ has been obtained by experimental study. Since $kp_i(t)$ is less than 1 at the start of the training process, CNN converges slower than CNNAR does, where from epoch 130, MSE falls. Consequently, more epochs are required to converge to a better solution. On the other and, Fig.4-(d) shows the performance measures of CNNAR with weight decay. After 300 epochs, the value of α grows, after which a noise is imposed on the network parameters to decrease the complexity of CNNAR. Furthermore, Fig.4-(e) reveals the performance measures of CNNAR when the coefficient of weight decay is $\alpha = 0.01$. This value has been derived from 10 times experiments. When the training process starts, the over-fitting does not occur and the learning algorithm is successful. However, when the training process continues, the over-fitting occurs in CNN and the weight decay term becomes completely effective.

To evaluate the different regularization schemes, in Table I, five different scenarios on CNN are compared including without regularization (WR), with adaptive dropout (AD), with constant dropout (CD), with adaptive weight decay (AW),

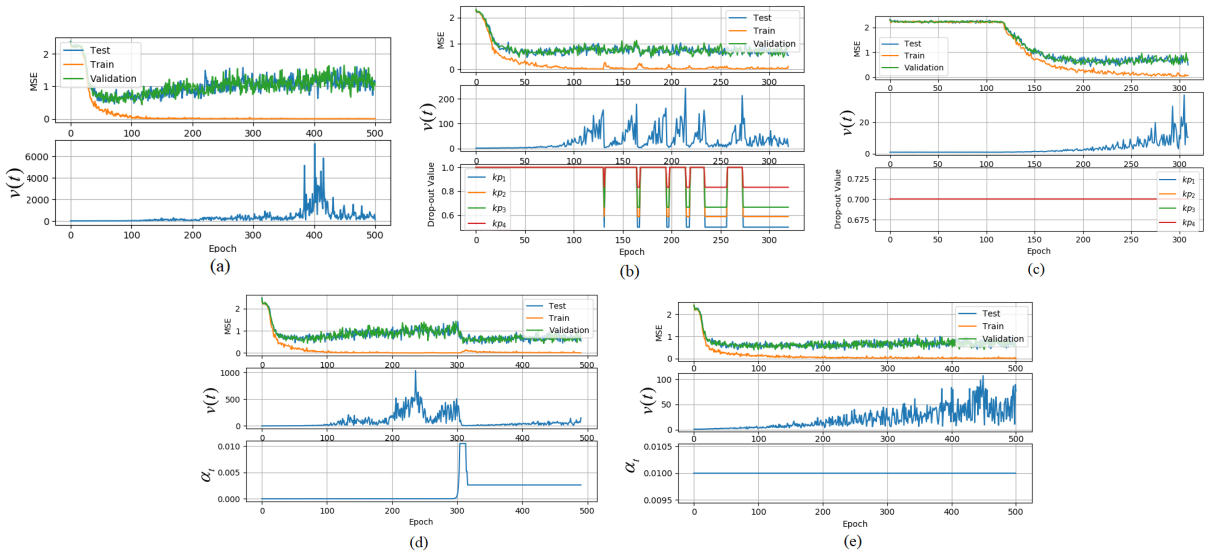

Fig. 4. Performances of CNN on SVHN dataset for (a): *WR*, (b): *AD,* (c): *CD*, (d): *AW*, and (e): *CW*.

TABLE I

PERFORMANCE ANALYSIS OF CNN BY CONSIDERING FIVE SCENARIOS ON HYPER-PARAMETERS

Dataset	Scenario	Precision	Recall	Accuracy	F-measure
SVHN	WR	81%	80%	82%	81%
	AD	91%	90%	91%	90%
	CD	84%	83%	84%	83%
	AW	90%	88%	90%	89%
	CW	84%	83%	84%	83%
CIFAR-10	WR	57%	56%	56%	56%
	AD	60%	60%	60%	60%
	CD	59%	57%	57%	56%
	AW	61%	61%	61%	61%
	CW	58%	58%	58%	58%
STL	WR	27%	28%	28%	27%
	AD	31%	29%	29%	29%
	CD	32%	26%	26%	27%
	AW	32%	31%	30%	30%
	CW	-%	-%	-%	-%

and with constant weight decay (CW). The precision, recall, accuracy, and f-measure values for three datasets have been evaluated by averaging 10 examinations. Note that in this study, we used 1/5 of samples of CIFAR-10 and 1/2 samples of STL datasets for the training process, since when we used the entire datasets, the over-fitting did not happen. Also, for classifying CIFAR-10 dataset, we used fractional max pooling which performed better on this dataset. In Table I, the CNN with AD has shown the best performance on SVHN, while CNN with AW has presented the best performance on both CIFAR-10 and STL datasets. In addition, the performance of CNN with AW has been close to that of CNN with AD in most cases. However, there is a great difference between CNNAR and CNN with constant regularization approaches. Indeed, CNNAR is at least 6% better than CNN with constant regularization on SVHN dataset. We have also found that in training CNN with constant weight decay on STL dataset, the learning algorithm did not converge. However, CNN with both of the adaptive regularizations converged successfully.

TABLE II

SENSITIVITY ANALYSIS ON THE ACCURACY OF CNNAR ON TRANSPORTATION-MODES (TMD) DATASET [41] WITH DIFFERENT TRAINING SAMPLES RATES IN 300 EPOCHS UNDER 5 SCENARIOS OF HYPER-PARAMETERS

Training samples rate	WR	CW	AW	CD	AD
15%	83.7%	83.2%	83.7%	84.1%	84.6%
45%	89.5%	89.1%	89.6%	89.6%	90.1%
75%	91.0%	91.0%	91.0%	91.0%	91.2%
Variance	10.66	11.7	10.46	9.45	8.60

B. Evaluation of Mode Detection

The transportation mode was detected in Ex-CADSE by a CNNAR. To this aim, TMD dataset [41] was used consisting of 1.376×10^6 samples of acceleration. The labels of these samples were bus, car, walking, train, and standstill. We only used accelerometer data of this dataset. Note that the accelerometer data were not calibrated and the samples were collected at different frequencies (less than 2 seconds). The averages of accuracies of CNNAR across 10 examinations under different training and testing ratios are presented in Table II. The last row of Table II, reports the variance of accuracies over all of the experiments on the CNN with different scenarios of hyper-parameters. Since the variance of CNN with AD is minimum while the accuracy values for all of the training sample rates are maximum, we can conclude that CNN with AD is a reliable and accurate classifier. By increasing the training sample rate, the over-fitting of the model decreases and when this rate is 75%, the model is not over-fitted.

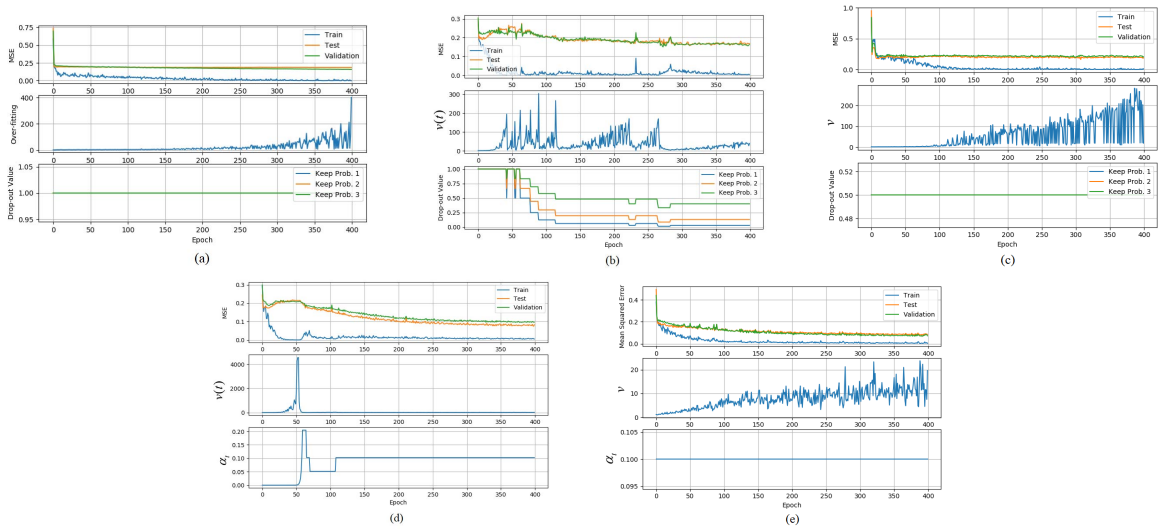

Fig. 5. Performances of CNN on *driving-styles dataset* for (a): *WR*, (b): *AD,* (c): *CD*, (d): *AW*, and (e): *CW*.

TABLE III

ACCURACY OF CNNAR ON TRANSPORTATION-MODES (TMD) DATASET COMPARED WITH THE OTHER METHODS BASED ON THE DIFFERENT SENSORS

Reference	Accuracy		Used Sensor(s)
	Motorized	Non-motorized	
CNNAR	**95.8**%	96.0%	Accelerometer
[29]	95.2%	**97.1**%	Accelerometer, Gyroscope, Magnetometer
[2]	77.8%	90.6%	GPS
[42]	85.0%	80.0%	Accelerometer
[43]	94.0%	95.6%	Accelerometer, GPS

TABLE IV

SEVEN CONFIGURATIONS OF CNNAR FOR CLASSIFICATION ON DRIVING-STYLES DATASET

Index	Convolution Layers	Pooling	Fully-Connected
A	$(200, 2, 1, 16)$ $(100, 1, 16, 32)$ $(25, 1, 32, 64)$	$(1, 4, 2, 1)$	$(1216, 512, 256, 128, 1)$
B	$(200, 2, 1, 16)$ $(100, 1, 16, 32)$ $(25, 1, 32, 64)$	$(1, 4, 2, 1)$	$(1216, 512, 256, 16, 1)$
C	$(300, 2, 1, 4)$ $(100, 1, 4, 8)$ $(30, 1, 8, 16)$	$(1, 4, 1, 1)$	$(608, 128, 32, 1)$
D	$(400, 2, 1, 8)$ $(100, 2, 8, 16)$ $(25, 2, 16, 32)$	$(1, 4, 1, 1)$	$(608, 512, 32, 1)$
E	$(200, 2, 1, 4)$ $(100, 2, 4, 8)$ $(25, 2, 8, 16)$	$(1, 4, 1, 1)$	$(608, 256, 32, 1)$
F	$(200, 2, 1, 4)$ $(100, 2, 4, 8)$ $(25, 2, 8, 16)$	$(1, 4, 1, 1)$	$(608, 1024, 64, 1)$
G	$(600, 2, 1, 4)$ $(200, 2, 4, 8)$ $(40, 2, 8, 16)$	$(1, 4, 1, 1)$	$(608, 768, 32, 1)$

For the next experiments, assume that we need to distinguish between two motorized and non-motorized classes by CNNAR. Then, we can merge the car, bus, and train classes into a single motorized class. The others are non-motorized modes. Now, we train CNNAR on 75% of the samples of this dataset. The CNN with AD performed the best. The best obtained accuracies were 95.8% and 96.0% for motorized and non-motorized classes, respectively. Table III, compares our results with other research findings. In this table, the precision of CNNAR model is very close to [1], but for motorized class, CNNAR outperforms the method of [1], while CNNAR applies only acceleration data. These differences suggest the preference of CNNAR to the previous machine learning techniques.

C. Evaluation of Driving Style

In this part, we discuss the performance of CNNAR for driving style evaluation by considering the previous five scenarios (WR, CD, AD, CW, and AW) (see Section V-A for details of scenarios). We consider a dataset described in [16]. Obviously, there are many configurations for a CNN model. Before experiment on CNNAR, we define seven configurations A-G as presented in Table IV. Then, in Tables V and VI, we evaluate the accuracies of different configurations of CNN with AD and AW, respectively. These tables reveal that, in all of the experiments, the precision, recall, and the f-measure have been better for normal drivers (ND) than the corresponding values for dangerous drivers (DD). The reason is that a dangerous driver sometimes performs normally, but more frequently drives dangerously. Thus, based on the normal driving in some time periods, we cannot be confident that the

TABLE V
COMPARISON OF CNNAR WITH AD ON SEVEN CONFIGURATIONS ON DRIVING-STYLES DATASET

Index	Precision		Recall		Accuracy		F-measure	
	DD	ND	DD	ND	DD	ND	DD	ND
A	75%	87%	77%	86%	83%		76%	87%
B	76%	90%	83%	86%	85%		80%	88%
C	78%	90%	83%	87%	86%		80%	89%
D	81%	86%	79%	89%	86%		80%	89%
E	74%	85%	73%	86%	81%		74%	85%
F	77%	84%	70%	88%	81%		72%	86%
G	81%	91%	83%	85%	88%		82%	90%

TABLE VI
COMPARISON OF CNNAR WITH AW ON SEVEN CONFIGURATIONS ON DRIVING-STYLES DATASET

Index	Precision		Recall		Accuracy		F-measure	
	DD	ND	DD	ND	DD	ND	DD	ND
A	93%	95%	87%	97%	93%		91%	95%
B	90%	98%	94%	96%	95%		93%	96%
C	89%	93%	87%	94%	92%		88%	93%
D	87%	95%	91%	93%	92%		89%	94%
E	78%	91%	85%	87%	86%		81%	89%
F	79%	95%	87%	91%	89%		85%	91%
G	83%	96%	89%	93%	91%		88%	93%

TABLE VII
COMPARISON BETWEEN CNNAR AND THE OTHER STUDIES ON DRIVING-STYLES DATASET

Reference	Accuracy	Approach	Used Sensor
Ex-CADSE	95.0%	overall	Accelerometer
CADSE [16]	94.0%	maneuver-based	Accelerometer, Magnetometer
[29]	91.1%	overall	Accelerometer, Gyroscope, Magnetometer
[8]	85.0%	maneuver-based	Accelerometer, GPS
[44]	90.5%	maneuver-based	Accelerometer, Gyroscope, Orientation, GPS
[7]	91.0%	maneuver-based	Accelerometer, Gyroscope, Orientation, GPS

driver is normal, which decreases the precision to recognize dangerous drivers.

In Fig. 5, indicates the training performance measures of CNN on driving-styles dataset [16] The training process without regularization on this dataset is shown in Fig.5-(a). Also Fig.5-(b)displays the performance of the CNN with AD. In this training process, the values of $kp_i(t)$ in each layer drops, when $\upsilon(t)$ grows. After reduction of $kp_i(t)$, the CNN weights are imposed by a shock which provides an opportunity for the learning algorithm to find a better weight for CNNAR. On the other hand, in Fig.5-(c), the training process with CD is shown where this approach cannot control the over-fitting of the model. Further, Fig. 5-(d) and (e) demonstrate the training process of CNN with AW and CW, respectively for configuration C. Fig.5-(d) indicates that when $\upsilon(t)$ exceeds $M_1(t)$, the value of α rises preventing the model from over-fitting. However, as can be observed in Fig.5-(e) the CW approach cannot prevent the model from over-fitting.

Finally, we compare the accuracy of CNNAR for driving style evaluation with that of other studies in Table VII. This comparison can be extended into two stages. Firstly, based on the utilized sensors, it can be shown that Ex-CADSE depends only on acceleration data, while the previous systems need more different types of sensor data. Secondly, based on the accuracy, Ex-CADSE is better than CADSE [16] which evaluates driving style with respect to the maneuvers. It is also better than [29] which evaluates the driving style overall. In general, Ex-CADSE provides the most accurate results and so it can be applicable to real world uses.

VI. CONCLUSION

Driving style evaluation is a challenging problem which depends on a huge volume of sensor data, type of data collection, context, environmental parameters, and driver's profile. However, capturing all of these concepts is very hard and we need to define some modern feature extractors and feature selectors on the collected data to provide the requirements for machine learning algorithms to finally recognize the driving style. The approach of this paper was use of a CNN for this purpose. However, there is not a simple way to choose optimal configuration and hyper-parameters of CNN. Also, the numerous parameters of CNN cause over-fitting through the training process. To resolve these problems, this paper focused on the regularization parameters. These hyper-parameters control the complexity of the learning model. We proposed two adaptive extensions of regularization called adaptive weight decay and adaptive dropout. In these extensions, when the generation power of the learning model diminished, the impact of regularization grew. In other cases, they increased the speed of the learning process. The reason is that, in the first step, the learning model is not over-fitted, then there is no need to utilize any regularization and so the adaptive regularization raises the speed of the classification process. By applying CNN with adaptive regularization, we extended CADSE system [16] to Ex-CADSE which detects the transportation mode and driving style continuously. This system evaluates the driving style overall and does not depend on maneuvers and so it can consider the effects of important features simply regarding traffic conditions, road surface, weather conditions, car stability, and car class on the driving style. Our experiments revealed that Ex-CADSE can detect the motorized mode with 95.8% accuracy and can classify the driving styles with 95.0% accuracy.

REFERENCES

[1] H. R. Eftekhari and M. Ghatee, "An inference engine for smartphones to preprocess data and detect stationary and transportation modes," *Transp. Res. C, Emer. Technol.*, vol. 69, pp. 313–327, Aug. 2016.
[2] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transp. Res. C, Emerg. Technol.*, vol. 86, pp. 360–371, Jan. 2018.

[3] R. B. Zadeh, M. Ghatee, and H. R. Eftekhari, "Three-phases smartphone-based warning system to protect vulnerable road users under fuzzy conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2086–2098, Jul. 2018.

[4] J. Wahlström, I. Skog, and P. Händel, "Smartphone-based vehicle telematics: A ten-year anniversary," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2802–2825, Oct. 2017.

[5] S. Friedman and M. Canaan, "Overcoming speed bumps on the road to telematics," Deloitte Univ. Press, 2014.

[6] R. Chhabra, S. Verma, and C. R. Krishna, "A survey on driver behavior detection techniques for intelligent transportation systems," in *Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng.-Confluence*, Jan. 2017, pp. 36–41.

[7] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1609–1615.

[8] C. Saiprasert, T. Pholprasit, and S. Thajchayapong, "Detection of driving events using sensory data on smartphone," *Int. J. Intell. Transp. Syst. Res.*, vol. 15, no. 1, pp. 17–28, Jul. 2015.

[9] P. Dhar, S. Shinde, N. Jadav, and A. Bhaduri, "Unsafe driving detection system using smartphone as sensor platform," *Int. J. Enhanced Res. Manage. Comput. Appl.*, vol. 3, pp. 65–70, Mar. 2014.

[10] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Driver behavior profiling using smartphones: A low-cost platform for driver monitoring," *IEEE Intell. Transp. Syst. Mag.*, vol. 7, no. 1, pp. 91–102, 2015.

[11] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2012, pp. 234–239.

[12] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2330–2339, Sep. 2017.

[13] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. González, "Safe driving using mobile phones," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1462–1468, Sep. 2012.

[14] F. Li, H. Zhang, H. Che, and X. Qiu, "Dangerous driving behavior detection using smartphone sensors," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1902–1907.

[15] H. Zhao, H. Zhou, C. Chen, and J. Chen, "Join driving: A smart phone-based driving behavior evaluation system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 48–53.

[16] M. M. Bejani and M. Ghatee, "A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 303–320, Apr. 2018.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[18] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech Language Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2015.

[19] W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang. (2016). "Characterizing driving styles with deep learning." [Online]. Available: https://arxiv.org/abs/1607.03611

[20] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[21] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 871–882, Jun. 2013.

[22] A. Jahangiri and H. A. Rakha, "Applying machine learning techniques to transportation mode recognition using mobile phone sensor data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2406–2417, Oct. 2015.

[23] X. Ma, Y. Li, Z. Cui, and Y. Wang. (2018). "Forecasting transportation network speed using deep capsule networks with nested LSTM models." [Online]. Available: https://arxiv.org/abs/1811.04745

[24] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[25] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[26] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959.

[27] J. Larsen, L. K. Hansen, C. Svarer, and M. Ohlsson, "Design and regularization of neural networks: The optimal use of a validation set," in *Proc. 6th Neural Netw. Signal Process.*, Sep. 1996, pp. 62–71.

[28] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.

[29] H. R. Eftekhari and M. Ghatee, "Hybrid of discrete wavelet transform and adaptive neuro fuzzy inference system for overall driving behavior recognition," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 58, pp. 782–796, Oct. 2018.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[31] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[32] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[33] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[34] S. K. Kumar. (2017). "On weight initialization in deep neural networks." [Online]. Available: https://arxiv.org/abs/1704.08863

[35] S. Koturwar and S. Merchant. (2017). "Weight initialization of deep neural networks (DNNS) using data statistics." [Online]. Available: https://arxiv.org/abs/1710.10570

[36] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[37] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. Comparison of overfitting and overtraining," *J. Chem. Inf. Comput. Sci.*, vol. 35, no. 5, pp. 826–833, 1995.

[38] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Hoboken, NJ, USA: Wiley, 2013.

[39] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, vol. 2011, no. 2, p. 5.

[40] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[41] M. D. Felice. *Transportation Mode Detection*. [Online]. Available: http://cs.unibo.it/projects/us-tm2017/index.html

[42] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proc. 11th ACM Conf. Embedded Networked Sensor Syst.*, 2013, p. 13.

[43] S. Reddym, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. Sensor Netw.*, vol. 6, no. 2, pp. 13-1–13-27, Feb. 2010.

[44] J.-H. Hong, B. Margines, and A. K. Dey, "A smartphone-based sensing platform to model aggressive driving behaviors," in *Proc. 32nd Annu. ACM Conf. Hum. Factors Comput. Syst.* New York, NY, USA: ACM, 2014, pp. 4047–4056.

Mohammad Mahdi Bejani is currently pursuing the Ph.D. degree with the Department of Computer Science, Amirkabir University of Technology. He is an expert Java Developer. He has published three papers in national and international journals and conferences. His research interests include intelligent transportation systems, data mining, and machine learning.

Mehdi Ghatee is currently an Associate Professor with the Department of Computer Science, Amirkabir University of Technology, Tehran, Iran. His majors include ITS, smartphone-based ITS systems, neural network, and fuzzy systems. He has published over 100 papers in national and international journals and conferences. He is currently the Associate Dean for Undergraduate Affairs with the Faculty of Mathematics and Computer Science and a member of the Board of Directors of ITS-RI and the Director of NORC.