A Data-Driven Misbehavior Detection System for Connected Autonomous Vehicles

ANKUR SARKER and HAIYING SHEN, University of Virginia, USA

In a Connected and Autonomous Vehicle (CAV) system, some malicious CAVs may send out false information in vehicle-to-vehicle communication to gain benefits or cause safety-related accidents. Previous false data detection methods are not sufficient to meet the accuracy and real-time requirements. In this paper, we propose a data-driven misbehavior detection system (MDS) (running by each CAV) that checks the consistency between the estimated and actually reported driving state (i.e., velocity, acceleration, brake status, steering angle) of an alerting CAV. *First*, MDS predicts the driving state using Gaussian mixture model based Mixture Density Network incorporating Recurrent Neural Network that can catch the driving behavior patterns of a CAV. *Second*, MDS extends the existing Krauss traffic flow model and uses it to consider the overall traffic flow of the road to make the predicted driving state more accurate. *Finally*, for a given received alert, a CAV validates the alert by checking the consistency between the predicted and actually reported driving states of the alerting CAV. We conduct extensive simulation studies based on a real driving dataset we collected from 29 participants and the *Simulator for Urban MObility* (SUMO) traffic simulator. The experimental results show that the false information detection rate of the proposed MDS is higher than other existing systems in different alert scenarios.

CCS Concepts: • **Information systems** → *Location based services*; • **Human-centered computing** → Ubiquitous computing; • **Computer systems organization** → Embedded and cyber-physical systems.

Additional Key Words and Phrases: Misbehavior detection system, Connected autonomous vehicles, Traffic flow model, Mixture density network, Alert validations.

ACM Reference Format:
Ankur Sarker and Haiying Shen. 2019. A Data-Driven Misbehavior Detection System for Connected Autonomous Vehicles. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 1 (January 2019), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Connected Autonomous Vehicles (CAVs) are autonomous vehicles with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication capacities. The fast-growing communication and sensing technologies enable CAVs to utilize more information to improve safety and reliability. The Society of Automotive Engineers (SAE) defined the Basic Safety Message (BSM) format that includes vehicle driving state (i.e., velocity (v), acceleration (a), brake status (b), steering angle (θ)) as shown in Fig. 1, and requires that each vehicle must send BSM to its nearby vehicles periodically (e.g., at every 0.1s) in V2V communication. Thus, it enables to introduce a lot of safety-related applications (e.g., emergency

This research is supported in part by U.S. NSF grants CCF-1822965, OAC-1724845, ACI-1719397, and CNS-1733596 and Microsoft Research Faculty Fellowship 8300751.
Authors' address: Ankur Sarker, as4mz@virginia.edu; Haiying Shen, hs6ms@virginia.edu, University of Virginia, Department of Computer Science, Charlottesville, VA, 22903, USA.

2474-9567/2019/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Driving state: [velocity (v), acceleration (a), brake status (b), steering wheel angle (θ)]

Fig. 1. Driving state of a vehicle.

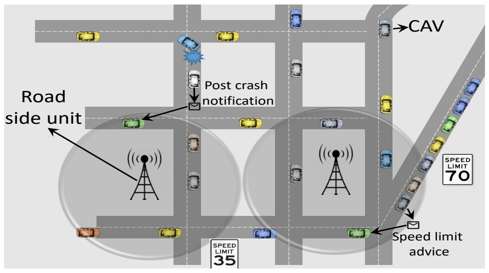


Fig. 2. Alert scenarios in a connected autonomous vehicles system.

electronic brake light, post crash notification, stopping/slow vehicle advising, and cooperative collision warning) [13, 14, 23, 24, 27]. As shown in Fig. 2, a CAV can send a *post crash notification* to its nearby CAVs and also send a *speed limit advice* to other CAVs approaching the same road section. However, such safety-related applications are vulnerable to false information attacks and may fail to provide appropriate services or lead to driving disruption or even fatal crashes. For example, an attacker vehicle B sends a false emergency electric brake light alert to its follower vehicle C, which may stop suddenly and leads to fatal rear-end collision with its following vehicle. Then, there arises a ***challenge:*** *how to design a system such that CAVs can successfully detect the false information from malicious (or compromised) CAVs in critical scenarios that require to take actions in real time (i.e., within next a few seconds)*?

There are several works [6, 8, 9, 12, 20, 22, 28] proposed for data-driven misbehavior detection in vehicular ad-hoc networks where vehicles usually misbehave by sending false information of itself and/or traffic condition. In these works, CAVs check whether the information sender can be at the current location based on the location of its reported event previously or whether it could be at the location of its reported event previously based on its current location. If the majority of the CAVs agree on the deviation value of the estimated location from the reported (or actual) location, the sender node is considered as misbehaving. However, these methods can only estimate the coarse location (i.e., a road section) of a vehicle to check if it is at the scene of its reported accident, so they cannot be applied to critical scenarios that require misbehavior detection action taking in a few seconds. The method in [22] could handle the critical scenarios. It utilizes the message-sending-time in the message from the sender CAV and message transmission time to estimate the location of the sender and then checks whether it is the same as its reported location. However, the sender can send false message-sending-time to make it consistent with its reported false location. Also, this method assumes that the vehicle velocity does not change, so it cannot cope with the velocity change or traffic flow rate change.

In this paper, we propose a data-driven misbehavior detection system (MDS) to more accurately detect the false information for safety-related applications for the CAVs. An alert receiver CAV runs MDS to validate the received alert information from an alert sender CAV. To detect the false information from an alert sender in real time, MDS utilizes the alert sender's previous driving states and locations and considers the overall traffic flow to predict its current driving state, and checks the consistency between the predicted and reported driving states. In addition, it considers the subsequent actions (i.e., driving states in the next a few seconds) of the alert sender and its nearby CAVs to check the truthfulness of the received information.

First, MDS uses the Gaussian Mixture Model (GMM)-based Mixture Density Network (MDN) incorporating a Recurrent Neural Network (RNN) to predict the driving state of the alert sender at the current time. *Second*, to more accurately predict the driving state of the alert sender, MDS uses the Krauss traffic flow model [16] to further consider the velocities of and inter-vehicular distances with its nearby CAVs to improve the prediction accuracy. *Finally,* MDS validates the alert from the alert sender by checking the consistency of the predicted and reported driving states and locations, and the consistency of the subsequent actions of itself and its nearby CAVs.

The following lists the major contribution of this paper:

(1) **Driving state prediction.** We build a Gaussian Mixture Model (GMM)-based Mixture Density Network (MDN) incorporating a Recurrent Neural Network (RNN) to predict the driving state of the alert sender at the current time based on its driving states in previous time points. MDN can predict the variances of input data with high accuracy. Since driving state data is a time series and usually is normally distributed, we chose RNN and GMM. RNN is able to derive various periodicity from a time series of input data, and GMM is more suitable to predict the output based on the given normally distributed input data.

(2) **Traffic flow consideration.** MDS then uses the predicted driving state predictions in Krauss traffic flow model [16], a well-known comprehensive traffic flow model which calculates vehicle movements considering the presence of other vehicles nearby, to improve prediction accuracy. We also improve the model by additionally use the predicted velocity deviation as an input for more accurate prediction. As a result, it can accurately estimate the driving state of an alert sender in a road section considering its velocity deviation and current states of its nearby vehicles.

(3) **Fake information detection mechanism.** We use a list of representative alerts (i.e., emergency electric brake light, road hazard notification, change of lane, emergency vehicle approach, and blind spot warning) as the examples. To detect the false alert provided by a CAV, MDS checks the inconsistency of the predicted and reported driving states and locations of the CAV and further considers its nearby CAVs' states and their subsequent actions. For each type of alerts, MDS checks certain conditions. In this way, MDS promptly detects the false alerts provided by CAVs in the critical traffic scenarios.

(4) **Extensive experimental evaluations.** We conduct extensive simulation studies based on a real driving dataset and the Simulator for Urban MObility (SUMO) microscopic traffic simulator [15] to evaluate the proposed MDS compared with existing methods. The real-world driving dataset from 29 participants is used to simulate and evaluate different scenarios. The simulation results show that MDS achieves higher detection accuracy than other existing methods.

The rest of the paper is organized as follows. Section 2 presents the background in this paper. Section 3 presents the system design of the proposed MDS. Section 4 evaluates MDS through extensive simulation studies. Section 5 discusses the existing literature. Finally, Section 6 concludes this paper with remarks on future work.

2 BACKGROUND

In this section, we first present the preliminaries and the underlying assumptions of our proposed system. We then introduce the MDN model and the Krauss traffic flow model used by MDS. Finally, we discuss different attack scenarios considered this paper.

2.1 Preliminaries

Suppose there are several CAVs driving on a section of a road network and each CAV is equipped with an On-Board Unit (OBU) to do computation, a Global Positioning System (GPS) unit to identify its location, and a communication unit to send messages to its nearby CAVs. When CAV i sends out a

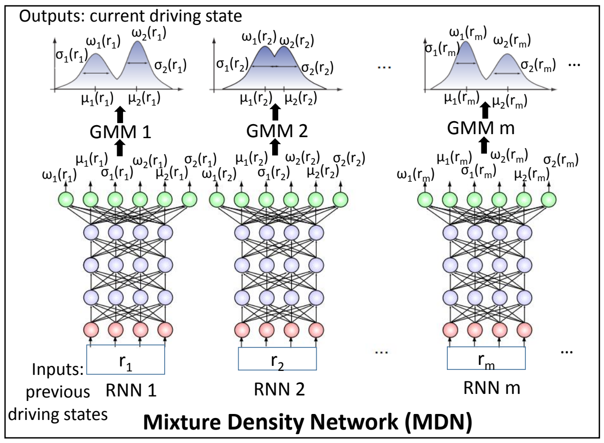

Outputs: current driving state
GMM 1
GMM 2
GMM m
Inputs: previous driving states
RNN 1
RNN 2
RNN m
Mixture Density Network (MDN)

Fig. 3. Mixture Density Network [29].

messages, only those CAVs inside the communication range of CAV i can receive the message. Each CAV i periodically sends to other CAVs the BSM containing current time stamp t and its own driving state (including velocity $v_i(t)$, acceleration $a_i(t)$, brake status $b_i(t)$, steering wheel angel $\theta_i(t)$), current location $l_i(t) = (x_i(t), y_i(t))$, heading direction, and vehicle dimension (i.e., length and width). In addition, an alert sender i raises an alert to warn other CAVs about some abnormal condition (e.g., traffic accident) or its sudden action change (e.g, lane change). When CAV j receives an alert from CAV i, MDS running in the OBU of CAV j needs to decide if the alert is false or not. MDS is mainly data-driven and it uses received information to check the consistency of the received alert with the predicted current driving state of sender CAV i and its nearby CAVs' driving states.

2.2 Assumptions

We have the following assumptions in our system:
(1) CAVs only send BSM (specified by SAE) and different alert messages to other CAVs. Also, CAVs do not share other information (e.g., traffic flow) with other CAVs. CAVs do not cooperate with other CAVs for misbehavior detection. Therefore, our approach is immune to the Sybil attacks where two or more vehicles collude to send false information together [22].
(2) We consider two types of alerts based on the motivation behinds misbehavior: active alerts and passive alerts [12]). In active attacks, the malicious vehicles try to gain some benefits directly without causing any physical harm (e.g., moving ahead, acquiring space). In passive attacks, the malicious vehicles do not try to gain some benefits directly but may cause physical harm (e.g., taking alliterative longer paths, causing rear-end collision).
(3) We do not rely on in-vehicle sensors (e.g., infrared or radar) attached to CAVs and our work is mainly for the V2V communication scenarios.

2.3 Mixture Density Network

The Mixture Density Network (MDN) is a machine learning model and it can represent any conditional probability distributions to describe any random functions [7]. MDN can predict the variances of input data and produce higher prediction accuracy than other machine learning techniques. We chose Recurrent Neural Network (RNN) and Gaussian Mixture Model (GMM) to build an MDN considering the features of driving state data: i) it is a time series, and ii) it is normally distributed [19]. The benefit of RNN over other machine learning methods (e.g., Hidden Markov Model, Kalman Filtering) is that RNN is able to derive various input patterns and cycles from a time series of input data. Other methods consider all input data equally and cannot detect the input patterns and cycles. GMM is more suitable to predict the output based on the given normally distributed input data.

As shown in Figure 3, the MDN combines a group of M number mixture models; each mixture model combining an RNN neural network and a GMM. The GMM is a probabilistic model that assumes all the data points are generated from a finite number of Gaussian distributions with unknown parameters. Each GMM has a Gaussian kernel function. From a set of input features $\mathbf{r}$, this GMM-based MDN learns the parameters of GMMs and outputs a probability density function of output features s, represented as $P(s|\mathbf{r}, M)$. MDN can obtain a sum of M Gaussian models with different means and standard deviations for different models. The conditional probability of a particular output $s(t)$ given the model state $\mathbf{r}(t)$ can be expressed by the following equation:

$$P(S = s|R = \mathbf{r}, M) = \sum_{m=0}^{M} w_m(\mathbf{r})\phi(s, \mu_m(\mathbf{r}), \sigma_m^2(\mathbf{r})) \tag{1}$$

where $\phi(s, \mu_m(\mathbf{r}), \sigma_m^2(\mathbf{r}))$ is the mth Gaussian kernel function, $\mu_m(\mathbf{r})$ and $\sigma_m^2(\mathbf{r})$ are the mean and variance vector of the mth Gaussian kernel function, respectively, and $w_m(\mathbf{r})$ are the weights of the Gaussian kernel functions. Training of the MDN model aims to maximize the log likelihood of the conditional probability $P(s|\mathbf{r}, M)$ as follows:

$$\hat{P} = \arg\max \sum_{n=1}^{N} \log P(s^n|\mathbf{r}^n, M) \tag{2}$$

where $\hat{P}$ is the maximum value of probability function $P(s|\mathbf{r}, M)$, N is total number of training samples. Once the training is done, we can obtain all the probability coefficients w_m, μ_m, and σ_m given some model state $\mathbf{r}(t)$.

2.4 Krauss Traffic Flow Model

In this paper, we use the well-known Krauss traffic flow model [16] to calculate the velocity of a vehicle based on the velocities and inter-vehicular distances of nearby vehicles. The Krauss traffic flow model takes the previous velocities, accelerations, and locations of a group of nearby vehicles as inputs and produces their current velocities and locations as outputs. The velocity and acceleration or deceleration of a vehicle based on each time period τ (e.g., 0.1 second) are $\delta v = \frac{\delta l}{\tau}$ and $\delta a = \frac{\delta v}{\tau}$, respectively, where l is its current location. We use t to denote the discrete time $t = n\tau$, $n = 0, 1, 2, ...$ Based on the model, the velocity of a vehicle at the next time $t + 1$ can be described as follows:

$$v_{t+1} = \min(v_{max}, \tilde{v}_{t+1}, v_{s,t}) \tag{3}$$

and the position of a vehicle at time $t + 1$ can be explained as follows:

$$l_{t+1} = l_t + v_{t+1}\tau \tag{4}$$

where v_t is the vehicle velocity at time step t; l_t is the vehicle co-ordinate; $\tilde{v}_{t+1}$ is the calculated velocity considering inter-vehicular distance; v_{max} is the maximum free flow velocity; $v_{s,t}$ is the minimum safety inter-vehicular velocity. Equation (3) shows that at each time stamp, a vehicle's velocity v_{t+1} depends on the free flow velocity v_{max}, synchronized inter-vehicular velocity $\tilde{v}_{t+1}$, safety vehicular velocity $v_{s,t}$.

In the following, we explain how to calculate $\tilde{v}_{t+1}$ and $v_{s,t}$ in Equation (3). Velocity $\tilde{v}_{t+1}$, is calculated based synchronized velocities of two neighbor vehicles (i.e., a vehicle and its the preceding vehicle) as

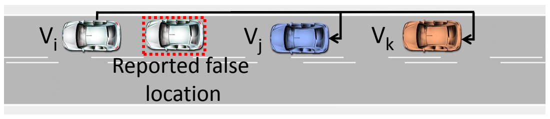


Fig. 4. False Emergency Electric Brake Light (EEBL) alert.

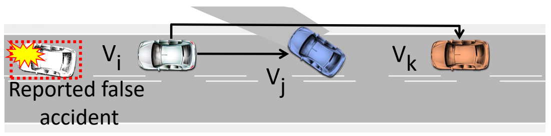


Fig. 5. False Road Hazard Notification (RHN) alert.

follows:

$$\tilde{v}_{t+1} = \begin{cases} v_{p,t}; d_t < D_t \\ v_t + a_t\tau; d_t \geq D_t \end{cases} \tag{5}$$

where $d_t = l_{p,t} - l_t - d_v$ is the inter-vehicular distance between the two neighbor vehicles, $l_{p,t}$ is the position of preceding vehicle, d_v is the vehicle length, and p represents the preceding vehicle, D_t is the synchronized inter-vehicular distance between these two neighbor vehicles, and it can be expressed as follows:

$$D_t = D(v_t, v_{p,t}), \tag{6}$$

$$D(u, w) = \lfloor k\tau u + a^{-1}\phi u(u - w) \rfloor \tag{7}$$

where $D(u, w)$ represents the inter-vehicular distance of two vehicles when their velocities are u and w, respectively, $k > 1$ and $\phi > 0$ are constants.

The safe inter-vehicular velocity $v_{s,t}$ in Equation (3) is calculated as follows:

$$v_{s,t} = \min(v_t^{safe}, \bar{v}_p^a - a\tau + \frac{d_t}{\tau}) \tag{8}$$

where $\bar{v}_p^a$ is the anticipated velocity of a vehicle based on its preceding vehicle p as follows:

$$\bar{v}_p^a = \min(v_{p,t}^{safe}, v_{p,t}, \frac{d_{p,t}}{\tau}) \tag{9}$$

$v_t^{safe} = \lfloor v^{safe}(d_{p,t}) \rfloor$ is the maximum safe inter-vehicle velocity of the Krauss model at time t [16] where $v^{safe}()$ calculates the maximum safe inter-vehicle velocity based on the preceding vehicle's velocity $v_{p,t}$ and its inter-vehicular distance from preceding vehicle at time t denoted by $d_{p,t}$. .

Overall, Equations (3)-(5) describe the velocity adaptation effect in a synchronized traffic flow (within the inter-vehicular distance range $d_{s,t} \leq d_t \leq D_t$, where $d_{s,t}$ is a safe inter-vehicular distance).

2.5 Alert Types and Misbehavior Scenarios

The alerts can be either observed or self-generated by the alert senders. As an example, a CAV might observe certain road features or hazard conditions (that influence vehicle driving) and generate an alert for other CAVs. Also, a CAV can generate an alert itself when it is decelerating rapidly or changing lane. There are three major groups of alerts presented based on the V2V communication systems: safety, convenience, and commercial purposes [5]. In this paper, we consider a list of five representative major alert types and misbehavior scenarios (indicated in [22]) as follows.

2.5.1 Emergency Electronic Brake Light (EEBL). The Emergency Electronic Brake Light (EEBL) notification enhances the safety of vehicles. It aims to avoid rear-end collisions when a preceding vehicle driving on the road suddenly applies brake, especially in dense driving situations.

Here, an attacker CAV i raises a false EEBL alert to notify its following CAVs that it is going to apply brake at current time t. This is a passive attack and it may cause damage (e.g., the following vehicles have rear-end collisions). As shown in Fig. 4, if CAV i really applies brake, its position should be at the location marked by red dotted squares denoted by x'. To avoid others from detecting its false alert, CAV i raises the alert and sends its false location $(x_{i'}, y_{i'})$ and velocity $v_{i'}$ inside the BSM. A smarter CAV i

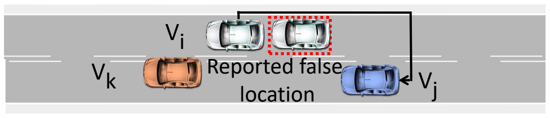


Fig. 6. False Change of Lane (CoL) alert.

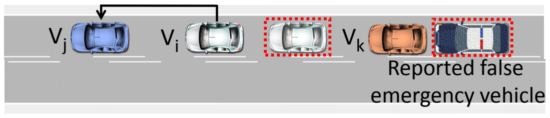


Fig. 7. False Emergency Vehicle Approaching (EVA) alert.

also sends a fake sending time stamp t' to prevent others from detecting the false alert by the message sending and transmission time [22]. Then, its following CAVs j and k would believe the received location information is correct [22] and the CAV i has applied brake already. The current positions of CAV i, CAV j, and CAV k are (x_i, y_i), (x_j, y_j), and (x_k, y_k), respectively where $(x_i, y_i) < (x_{i'}, y_{i'}) < (x_j, y_j) < (x_j, y_j)$ as shown in Fig. 4. CAV j would think it has to be stopped within $d_1 = (\sqrt{(xi' - x_j)^2 + (yi' - y_j)^2} + \delta_1)$ and CAV k would think it has to stopped within $d_2 = (\sqrt{(xi' - x_j)^2 + (yi' - y_j)^2} + \delta_2)$ where δ_1 and δ_2 are distances CAV i and CAV j would cover before making full stops, respectively. Here, if the difference between distances d_1 and d_2 is less than the distance between CAV j and CAV k, then CAVs j and k would have a rear-end collision together.

2.5.2 *Road Hazard Notification (RHN).*

Road Hazard Notification (RHN) includes hazardous traffic condition notification (e.g., tractor, horse-drawn buggy), traffic accident notification (e.g., post crash notification) and road feature/damage notification (e.g., potholes, a pedestrian or bicyclist on a narrow roadway, the presence of animals on or near roadway, fallen trees or other objects, slippery pavement due to ice or mud). A preceding CAV sends an RHN alert to its following CAVs, who are not able to observe the conditions and would take a detour if necessary.

Here, the attacker CAV i raises a false RHN alert along with its BSM to give false information to its following CAVs about the incoming road hazard at current time t. This is a passive attack without causing any physical damage but it may cause all the following CAVs to take alternative long paths. Suppose CAV i reports its current position as (x_i, y_i) and generates a false RHC alert about a false congestion d_i meters ahead of CAV i. Also, CAV i reports that it has already stopped at current time t as shown in Fig. 5. CAVs j and k are far away from CAV i but inside the communication range of CAV i.

2.5.3 *Change of Lane (CoL).*

Change of Lane (CoL) alert enhances the safety of vehicles in a dense driving environment. It aims to avoid fatal collisions which can occur if a vehicle changes its current lane suddenly on a roadway.

Here, an attacker CAV i driving in one lane raises a false CoL alert (along with its BSM) so that it will be ahead of its following CAV j in another lane. This is an active attack where CAV i tries to gain the space in another lane and move ahead of CAV j in another lane. More specifically, CAV i at time t with velocity v_i sends a false CoL alert to other CAVs that it is going to change lane and its current position is $(x_{i'}, y_{i'})$ instead of its real position (x_i, y_i) as shown in Fig. 6. Because the short distance between CAV i and CAV k prevents CAV i from changing lane, the false reported location makes it appear that the inter-vehicle distance between CAV i and CAV k is large enough for the lane changing immediately. After receiving the alert, other nearby CAVs in another lane (e.g., CAV j) would try to give CAV i more space so that the safety inter-vehicle distanced is maintained where $(x_i, y_i) < (x_{i'}, y_{i'}) < (x_j, y_j)$.

2.5.4 *Emergency Vehicle Approaching (EVA).*

Emergency Vehicle Approaching (EVA) alert notifies the preceding CAVs that there is an emergency vehicle approaching. After receiving the alert, the preceding CAVs would try to create space to let the emergency vehicle pass smoothly and quickly.

In this scenario, an attacker CAV i raises a false EVA alert to its preceding CAV j so that it can move ahead of CAV j. This is an active attack since CAV i tries to move ahead of its preceding CAV j. More specifically, CAV i reports to its preceding CAV j about its false current stopping location $(x_{i'}, y_{i'})$ and false emergency vehicle location (x_e, y_e) at current time t as shown in Fig. 7.

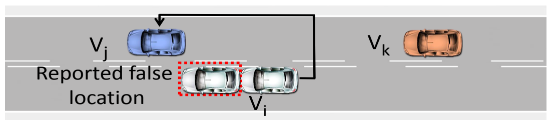


Fig. 8. False Blind Spot Warning (BSW) alert.

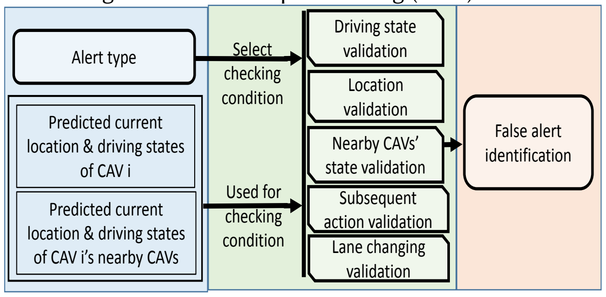

Alert type
Predicted current location & driving states of CAV i
Predicted current location & driving states of CAV i's nearby CAVs
Select checking condition
Used for checking condition
Driving state validation
Location validation
Nearby CAVs' state validation
Subsequent action validation
Lane changing validation
False alert identification

Fig. 9. The overall architecture of the proposed system.

2.5.5 Blind Spot Warning (BSW). CAV i sends out a Blind Spot Warning (BSW) alert to notify its preceding CAV j in a neighboring lane that it is in the blind spot of CAV j and CAV j should not change lane until its blind spot is clear. This alert aims to avoid collision between two CAVs during lane changing, especially in the snowy or foggy weather.

Suppose CAV j tries to change lane at current time t and CAV i sends false BSW alert with its velocity v_i and false location $(x_{i'}, y_{i'})$ (that is the blind spot of CAV j) instead of its current location (x_i, x_j) as shown in Fig. 8. Here, CAV i can easily estimate the blind spot location of CAV j based on CAV j's BSMs. After receiving the alert, CAV j stops changing lane immediately and meanwhile attacker CAV i can move forward. This is an active attack since attacker CAV i tries to prevent other CAVs from being in front of it so it can move forward fast.

3 SYSTEM DESIGN

3.1 Overview of the Misbehavior Detection System (MDS)

In this paper, we propose a Misbehavior Detection System (MDS) to enable a CAV to validate its received alert from another CAV. As shown in Fig. 9, for a given alert type, MDS uses the driving states of an alert sender and its nearby CAVs to validate the alert. It mainly validates the location and driving states of an alert sender and its nearby CAVs, and their subsequent actions. The CoL alert needs to be handled specifically since it has different features from other alerts. *First*, MDS uses the previous driving states of an alert sender (retrieved from its BSMs) to predict its current driving state (Section 3.2). *Second*, MDS estimates the movements of all CAVs surrounding the alert sender. Traffic flow in an inter-section mostly depends on the interactions between nearby vehicles (Section 3.2.2). *Third*, based on the nature of the alerts, MDS needs to check a few actions of an alert sender and its nearby CAVs to validate the received alert (Section 3.3). We present the details of these step in the following.

3.2 Driving State Prediction

3.2.1 Machine Learning Based Driving State Prediction. Recall that the driving state of a CAV is represented by $[v(t), a(t), b(t), \theta(t)]$. MDS predicts the current driving state ($s(t) = [v(t), a(t), b(t), \theta(t)]$) of an alert sender based on its previous driving states ($\mathbf{r}(t) = [r(t-1), r(t-2), ..., r(t-T)]$) obtained from the received BSMs of the alert sender. Using MDN, we aim to give a representation of the conditional probability $P(s = S|\mathbf{r} = R)$. Using $\mathbf{r}(t)$ as input feature, we can predict the current state $s(t)$ as output

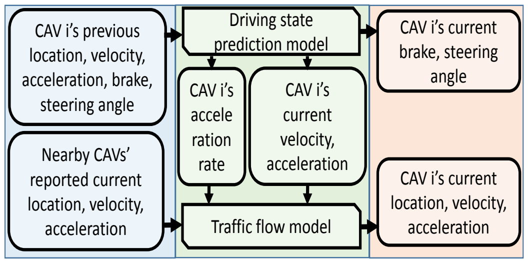

CAV i's previous location, velocity, acceleration, brake, steering angle
Nearby CAVs' reported current location, velocity, acceleration
Driving state prediction model
CAV i's accele ration rate
CAV i's current velocity, acceleration
Traffic flow model
CAV i's current brake, steering angle
CAV i's current location, velocity, acceleration

Fig. 10. Driving state and location validations.

feature of the MDN model (as described in Section 2.3). The MDN model utilizes a set of mixture models to provide the conditional probability distribution of $P(s = S|\mathbf{r} = X)$.

A mixture model uses a neural network to extract the structural features of the input states. Instead of using the Convolutional Neural Network, we use an RNN as the neural network because RNN is able to extract various periodic features (i.e., various input patterns and cycles) from a time series of previous driving states. An RNN takes into account multiple previous states as inputs rather than just single state $([r(t-1), r(t-2), ..., r(t-T)])$ and the output of the RNN is the current state $([s(t)])$. Different CAVs driving styles (i.e., applied acceleration, brake pressure, and steering wheel movements) are different and the RNN is able to find the periodicity of and non-linear relationships of these given driving states [18].

The correlation between simultaneous correlated behaviors (including acceleration, brake status, and steering angle) for a CAV should keep similar in driving. To measure the correlation, we calculate the joint probability distribution of acceleration, brake status, and steering angle from the predicted driving state s:

$$f_{a,b,\theta}(a_t, b_t, \theta_t) = F_a(a_t)F_b(b_t)F_\theta(\theta_t) \tag{10}$$

$F_a(a_t)$, $F_b(b_t)$, and $F_\theta(\theta_t)$ are the cumulative distribution functions which can be calculated from the predicted mean and variance as given by MDN model.

3.2.2 *Traffic Flow Consideration.*

In the above, we predict the driving state of a CAV only based on its previous driving states. However, the prediction is not accurate enough without the consideration of its nearby CAVs' driving states. Thus, as shown in Fig. 10, to more accurately predict the location, velocity, and acceleration of a CAV, MDS uses the Krauss traffic flow (Section 2.4) to additionally considers the velocities and inter-vehicular distances of its nearby CAVs.

As explained in Section 2.4, for the velocity estimation, the Krauss traffic flow model considers several factors including speed limit, free flow velocity of the road section, safe inter-vehicular velocity and distance, and synchronized traffic flow based on the nearby CAVs. However, it is not sufficiently accurate as it does not consider velocity deviation [19]. MDS further improves the Krauss traffic flow model by taking into account the historical velocity deviation (which is calculated using the MDN model as described in Section 3.2) as follows. Then, based on Equation (3), the velocity of an alert sender CAV can be described with the consideration of velocity deviation ε_t of that CAV as follows:

$$v_{t+1} = \min(v_{max}, \tilde{v}_{t+1} + \varepsilon_t\tau, v_{s,t}) \tag{11}$$

Here, we consider the velocity deviation ε_t in Equation (11) which simulates the acceleration or deceleration rate based on the CAV movements as follows:

$$\varepsilon_t = \begin{cases} -a\delta t; & \tilde{v}_t < v_t \\ a\delta t; & \tilde{v}_t > v_t \\ \eta_0; & \tilde{v}_t = v_t \end{cases} \tag{12}$$

where a is acceleration calculated from previous driving states using Equation (1). η_0 is the randomly chosen value from (-1,1). Thus, each CAV i periodically calculates the velocities of all nearby CAVs based on the Equation (11) with the help of current velocity deviation ε_t.

3.3 Validations of Different Alerts

The MDS running in an alert receiver checks the truthfulness of a received alert by checking several conditions together. Given an alert type, it checks certain conditions correspondingly. In this paper, we identified several major checking conditions and more conditions can be added. Below, we first present the checking conditions and then explain how MDS checks the truthfulness of each type of alerts.

3.3.1 Driving State Validation. An alert receiver validates the reported driving state of the alert sender CAV at time t based on its previous states (Fig. 10) using MDN model and traffic flow model.

First, the alert receiver calculates the expected velocity of an alert sender CAV at time t based on its previous velocity and its nearby CAVs' velocities at time $(t-1)$ using Equation (11), which uses the estimated velocity, velocity deviation and acceleration from MDN. Then, MDS calculates the difference between the predicted velocity and reported velocity and if the difference is greater than a certain threshold, there is a mismatch and the reported velocity is not correct. Second, we check the correlation of the reported values of acceleration, brake, and steering angle values with respects to ($w.r.t.$) their predicted values using their joint probability distribution calculated by Equation (10). If the value is less than a threshold, we can say that the reported acceleration, brake status, and steering angle are not correct. In some alert scenarios (i.e., CoL and BSW), an alert sender's previous historical driving states must be consistent with its current driving state. In these scenarios, MDS applies the paired t-test, which is a statistical hypothesis test [10], to compare current time window (e.g., 1 second) with the most recent time window of received data to check if these consecutive two time windows are from similar distribution. A t-test proves the null hypothesis that the difference between two time windows of received data measured on the same statistical unit (e.g., mean) has a mean value of zero. The t-test for the received acceleration data can be evaluated as follows:

$$\mathcal{T}_a = \frac{\bar{a_t} - \mu_{a_{t-1}}}{\sqrt{\frac{s^{a,t}}{n_1} + \frac{\sigma^2_{a,t-1}}{n_2}}} \tag{13}$$

where $\mathcal{T}_a$ is magnitude of the t-test for currently received acceleration data, $\bar{a_t}$ is the mean of the currently received acceleration data, $\mu_{a_{t-1}}$ is the mean of acceleration collected previously, $s_{a,t}$ is the standard deviation of currently received sample acceleration, $\sigma^2_{a,t-1}$ is the standard deviation of previously received acceleration data, n_1 and n_2 are the total number of times of received acceleration data from current and previous time windows, respectively. The magnitude of t-test $\mathcal{T}_a$ should be greater than a certain threshold if two time windows of received acceleration data come from the same distribution which means current acceleration is similar to the previous acceleration. Thus, MDS validates the similarity of currently received acceleration data with the most recent historical acceleration data to validate the received alert. Using the same way, MDS can calculate $\mathcal{T}_v$, $\mathcal{T}_b$, and $\mathcal{T}_\delta$ in order to validate velocity, brake status, and steering angle, respectively.

3.3.2 Location Validation. Next, a CAV validates the currently reported location based on the previous location information of the alert sender. In most cases, the reported location of an alert sender at time t is always consistent with its location at time $(t-1)$. To validate the reported position, MDS predicts the location of an alert sender CAV i at time t ($(x_{i,t}, y_{i,t})$) based on its previous location $(x_{i,t-1}, y_{i,t-1})$, velocity $v_{i,t-1}$, steering angle $\theta_{i,t-1}$ at time $(t-1)$ as follows:

$$x_{i,t} = x_{i,t-1}\delta t + v_{i,t-1}\cos\theta_{i,t-1} \tag{14}$$

$$y_{i,t} = y_{i,t-1}\delta t + v_{i,t-1}\sin\theta_{i,t-1} \tag{15}$$

In some alert scenarios (e.g., EEBL), an alert sender CAV i does not necessarily follow its previously reported velocity $v_{i,t-1}$. In this case, MDS calculates velocity $v_{i,t-1}$ using current velocity $v_{i,t}$ and acceleration $a_{i,t}$ (i.e., $v_t = v_{t-1}\delta t + a_t$). Then, MDS predicts location $(x_{i,t}, y_{i,t})$ using the above equations. MDS compares the predicted location $(x_{i,t}, y_{i,t})$ with the received location $(x'_{i,t}, y'_{i,t})$. If the two locations are close (i.e., less than the GPS positioning error), MDS concludes the reported location $(x'_{i,t}, y'_{i,t})$ is valid.

3.3.3 Nearby Vehicles' State Validation. In the congested roadway, the velocity of an alert sender CAV i is mostly affected by the velocity of its preceding CAV j. Thus, an alert receiver needs to check the consistency of CAV i's current velocity and location $w.r.t.$ the current velocity and location of its preceding CAV or its following CAV. Suppose there are two CAVs driving in a single lane where CAV i follows CAV j. Based on the velocity of CAV j, CAV i's velocity is adjusted over time. We can describe the velocity of CAV i at time time t, $v_{i,t}$ from Equation (11) as follows:

$$v_{i,t} = \begin{cases} v_{i,t-1} + a_t\tau_i + \varepsilon_{i,t}; d_{t-1} \geq D_{t-1}, v_{i,t-1} < v_{max} \\ v_{j,t-1} + \varepsilon_{i,t}; \qquad d_{t-1} < D_{t-1} \\ v_{max} + \varepsilon_{i,t}; \qquad otherwise \end{cases} \tag{16}$$

where d_{t-1} is the space between CAV i and CAV j at time $(t-1)$, D_{t-1} is the synchronized inter-vehicular distance, $v_{j,t-1}$ is the velocity of CAV j at time $(t-1)$, and $\varepsilon_{i,t}$ is the velocity deviation of CAV i. It means that the velocity of CAV i at time t approximately is equal to the velocity of its preceding CAV j at time $(t-1)$ when their inter-vehicular distance is less than the synchronized inter-vehicular distance. This relationship can be checked to find if one of the CAVs gives false current velocity.

MDS applies the same approach for checking the velocity relationship between an alert sender with its preceding CAV and with its following CAV. Suppose three CAVs j, i, and k are driving one after another in a single lane; CAV k follows CAV i that follows CAV j. Let us consider the velocity of CAV j at time $t-2$ as follows:

$$v_{j,t-2} = z + \varepsilon_{j,t-2} \tag{17}$$

where $z < v_{max}$ is constant and $\varepsilon_{j,t-2}$ is the velocity deviation of CAV j at time $t-2$ (as in Equation (12)). Then, based on the second case in Equation (16), if the inter-vehicle distances are less than the synchronized distance D, the velocities of CAVs i and k at time $t-1$ and t, respectively, can be described as follows:

$$v_{i,t-1} = (z + \varepsilon_{j,t-2}) + \varepsilon_{i,t-1} \tag{18}$$

$$v_{k,t} = (z + \varepsilon_{j,t-2} + \varepsilon_{i,t-1}) + \varepsilon_{k,t} \tag{19}$$

$\varepsilon_{i,t-1}$, and $\varepsilon_{k,t}$ can be calculated based on Equation (12). $v_{j,t-2}$, $v_{i,t-1}$ and $v_{k,t}$ are reported velocities from CAVs j, i, and k, respectively. From each equation, we can calculate the z value. We can compare two z values, and if they are different, one of the vehicles reported false velocity. This requires two consecutive messages. Assume no more than one of the three vehicles reported false velocity, then we can compare the three z values, and the CAV for the different z value is identified as the malicious vehicle.

Next, we check the reported locations of an alert sender CAV *w.r.t.* its nearby CAVs' locations using Equations (14) and (15). If there is a conflict between reported locations of one alert sender AV and its neighbor AV, MDS can conclude that at least one of the reported locations from these CAVs is not correct. The conflict here means an alert sender's location is after its following vehicle or is before its preceding vehicle, which is impossible. In the EVA alert scenario, since the alert sender's following CAVs slow down or stop earlier than the alert sender, an alert receiver can further check their relative locations to validate the alert. Let's consider the previous case where three CAVs j, i and k drive in a single lane. If the velocity of CAV i is $v_{i,t-1}$ at time $(t-1)$, and it is slowing down in constant rate δv_i, then we can describe its covered distance as follows:

$$d_t > d_{t-1}, \tag{20}$$

$$d_t = \big((v_{i,t-1} + \delta v_{i,t}) - (v_{k,t-1} + \delta v_{k,t})\big)\delta t + d_{t-1} \tag{21}$$

where d_{t-1} is the inter-vehicular distance between CAV i and CAV k and $\delta v_{i,t-1}$ is the unit velocity change at time $(t-1)$. It means that the distance covered by CAV i at time t is less than the distance covered by CAV k at time t due to the alert scenario.

3.3.4 Subsequent Action Validation. Additionally, an alert receiver needs to check the subsequent actions of an alert sender i and its nearby CAVs for a subsequent time period after receiving an alert at time t. Thus, MDS checks the subsequent velocities and locations of an alert sender i and its nearby CAVs for a short period of time based on the received few messages for a few time slots.

Here, MDS checks two types of the subsequent actions of an alert sender CAV i: increasing or decreasing velocity to adjust its velocity based on its preceding CAV (i.e., acceleration or deceleration) and making a complete stop. For both of the actions, we can check the consistency between the reported locations and velocities of the alert sender CAV and its preceding CAV and following CAV. Specifically, we use the same method introduced above that uses Equations (17), (18), and (19). Here, we need to check subsequent actions, i.e., calculate $v_{j,t}$, $v_{i,t+1}$, and $v_{k,t+2}$ at time t, $t+1$, and $t+2$, respectively for the alert validation. This validation method is also effective when the velocities of these CAVs are decreasing or increasing simultaneously.

In addition to velocity, we can also check the distance. If CAV i follows CAV j, then their inter-vehicular distance d_{t+1} at time $t+1$ should follow:

$$d_{t+1} = \left\lceil \frac{\big((v_{j,t+1} + v_{j,t}) - (v_{i,t+1} + v_{i,t})\big)\delta t}{2} \right\rceil \tag{22}$$

If the inter-vehicular distance d_{t+1} is not equal to the difference of reported locations from CAV i and CAV j, we can tell that one of two CAVs' reported information is not accurate. The number of messages needed to identify a malicious CAV is the same as the velocity validation in the above.

3.3.5 Lane Changing Validation. The CoL alert has special features compared with other types of alerts and needs to be validated specifically. To validate a CoL alert for an alert sender CAV i, an alert receiver needs to check if there is enough inter-vehicular distance in the target lane, and also if CAV i's current velocity is suitable to change lane based on the velocity of its nearby vehicles. We can derive the distance and velocity requirements of a lane changing event as follows:

Distance requirements. Before changing the current lane, CAV i needs to check the inter-vehicular distance between two neighbor CAVs in the target lane. The safety inter-vehicular distance condition for a lane changing event of CAV i at current time t is as follows:

$$d^p_{i,t} > \min(v_{i,t}\tau, D(v_{i,t}, v^p_{i,t})) \tag{23}$$

$$d^f_{i,t} > \min(v^f_{i,t}\tau, D(v^f_{i,t}, v_{i,t})) \tag{24}$$

where $d^p_{i,t}$ is the inter-vehicular distance between an alert sender i and its preceding CAV in the target lane at time t. Similarly, $d^f_{i,t}$ is the inter-vehicular distance between an alert sender i and its following CAV in the target lane, where $v_{i,t}$ is the velocity of CAV i at a time t and $v^f_{i,t}$ is the velocity of following CAV of in the target lane at time t. $D(v_{i,t}, v^p_{i,t}))$ can be calculated using Equation (6).

Equation (23) means that the inter-vehicular distance between alert sender i and its preceding CAV in the target lane at time t should be greater than the minimum of the distance covered by CAV i and their synchronized inter-vehicular distance at time t. Equation (24) means that the inter-vehicular distance between CAV i and its following CAV in the target lane at time t should be greater than the minimum of the distance covered by the following CAV and their synchronized inter-vehicular distance at time t. Therefore, MDS checks these two conditions to validate the lane changing event. However, if at least one of the above conditions is not satisfied then, the synchronized inter-vehicle distance of two neighbor vehicles is not large enough to change lane. Then, in order to make sure if the distance is enough for lane changing MDS further checks the following condition:

$$(l^p_{i,t} - l^f_{i,t}) > \lceil \lambda(v^f_{i,t} - v^p_{i,t})\delta t + d_v \rceil \tag{25}$$

where $l^p_{i,t}$ and $l^f_{i,t}$ are the positions of preceding CAV and following CAV of the CAV i in the target lane at time t, respectively. Equation (25) means that the intermediate distance between the preceding and following CAVs of CAV i in the target lane should be greater than the summation of the distance covered by the preceding and following CAVs of the CAV i in the target lane and the length of the CAV.

Velocity requirements. The velocity requirements of an alert sender i for moving from one lane to another lane at time t are as follows:

$$v^p_{i,t} \leq v_{p,t}, \tag{26}$$

$$v_{i,t} \leq v_{p,t} \tag{27}$$

where $v^p_{i,t}$ is the velocity of the preceding vehicle in the target lane and $v_{p,t}$ is the velocity of the preceding vehicle at time t. Equations (26) and (27) state that the velocity of CAV i's preceding CAV should be greater than or equal to the velocity of the CAV i and it should also be greater or equal to the velocity of the preceding CAV in the target lane, respectively. Otherwise, the lane changing action would reduce the traffic flow in the target lane. Therefore, MDS checks the above velocity requirements to validate the lane changing event.

3.3.6 Alert Validation. Table 1 summarizes the characteristics of different alerts for validation. The MDS running in an alert receiver checks the truthfulness of a received alert from alert CAV i. First, MDS conducts *location validation* and *velocity validation* in nearby vehicles' state validation since all types of

Table 1. Characteristics of different alerts.

Alerts	Alert sender's current states are similar to the previous states	CAVs whose state changes are similar to the alert sender	CAVs whose subsequent actions are similar to the alert sender	Alert receiver/validator
Emergency Electric Brake Light (EEBL)	No	Preceding CAV	Preceding CAV	Following CAV
Road Hazard Notification (RHC)	No	Preceding and following CAVs	Preceding and following CAVs	Following CAV (far away)
Change of Lane (CoL)	Yes	Preceding and following CAVs	Following CAV('s velocity increasing)	Following CAV (in another lane)
Emergency Vehicle Approaching (EVA)	No	Following CAV	Following CAV	Preceding CAV
Blind Spot Warning (BSW)	Yes	Preceding and following CAVs	Following CAV	Preceding CAV (in another lane)

alerts need these validations. Then, MDS considers other conditions to validate the received alert based on its characteristics.

ALGORITHM 1: Pseudocode for Alert verification method.

```
    Input: Driving state of a CAV $i$ and its nearby CAVs, and the received alert from CAV $i$
    Output: Received alert is valid or not ($Truthfulness$)
 1  $C_1 \leftarrow$ CAV $i$'s reported location validation (Equ. (14), (15))
 2  $C_2 \leftarrow$ CAV $i$'s the nearby CAVs' state validation (based on Equ. (16))
 3  if $Alert\ is\ EEBL$ then
 4      $C_3 \leftarrow$ subsequent action validation of the preceding CAV of CAV $i$ (Equ. (16), (22))
 5      $Truthfulness \leftarrow C_1 \& C_2 \& C_3$
 6  end
 7  else if $Alert\ is\ RHC$ then
 8      $C_3 \leftarrow$ subsequent action validation of the preceding and following CAVs of CAV $i$ (Equ. (16),(22))
 9      $Truthfulness \leftarrow C_1 \& C_2 \& C_3$
10  end
11  else if $Alert\ is\ CoL$ then
12      $C_3 \leftarrow$ subsequent action validation of the preceding and following CAVs of CAV $i$ (Equ. (16),(22))
13      $C_4 \leftarrow$ driving state of CAV $i$ validation (Equ. (10), (11), (13))
14      $C_5 \leftarrow$ lane changing of CAV $i$ validation (Equ. (23), (25),(26),(27))
15      $Truthfulness \leftarrow C_1 \& C_2 \& C_3 \& C_4 \& C_5$
16  end
17  else if $Alert\ is\ EVA$ then
18      $C_3 \leftarrow$ subsequent action validation of the following CAV of CAV $i$ (Equ. (21),(22))
19      $Truthfulness \leftarrow C_1 \& C_2 \& C_3$
20  end
21  else if $Alert\ is\ BSW$ then
22      $C_3 \leftarrow$ subsequent action validation of the following CAVs of CAV $i$ (Equ. (16),(22))
23      $C_4 \leftarrow$ driving state of CAV $i$ validation (Equ. (10), (11), (13))
24      $Truthfulness \leftarrow C_1 \& C_2 \& C_3 \& C_4$
25  end
26  return $Truthfulness$
```

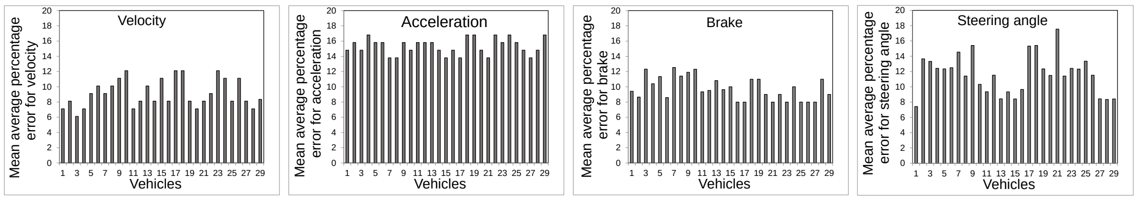
Algorithm 1 presents the pseudocode to validate different received alerts. Initially, it validates the reported location of a CAV and its nearby CAVs' states (Steps 1–2). Then, different alerts are validated based on their requirements (as shown in Table 1).

(1) If the received alert is EEBL, then it is validated using the location validation, nearby vehicles' state validation, and subsequent action validation (Steps 3–5). In an EEBL alert, only the following CAVs of the alert sender are affected and they should conduct alert validation and take action accordingly. In EEBL alert, the velocities of the preceding CAV of the alert sender and itself should be similar for that next several time stamps, the subsequent actions of the alert sender and its preceding CAV should be similar (e.g., decreasing velocity drastically).
(2) If the received alert is RHC, then it is validated using location validation, nearby vehicles' state validation, and subsequent action validation (Steps 6–8). In an RHC alert, only the following CAVs far away from the alert sender (that cannot see its surroundings) are affected and they should conduct alert validation and take action accordingly. Before and after the RHC alert, the locations and driving states of the alert sender and its nearby preceding and following CAVs should be similar (e.g., having similar velocities). Thus, the received RHC alert can be validated if the driving state of a CAV is consistent with its nearby CAVs' driving states.
(3) If the received alert is CoL, then the driving state validation, location validation, nearby vehicles' state validation, and subsequent action validation are conducted (Steps 9–13). In a CoL alert, only the following CAV of the alert sender in the target lane take is affected and it should conduct alert validation and take action accordingly. During lane changing, there should be enough space between the preceding CAV and following CAV in the target lane and the velocity of preceding CAV in the target lane should be greater or the same as the velocity of the alert sender. Thus, if these conditions are not satisfied, the received CoL alert is considered not valid. In addition, we can check the subsequent actions of the alert sender and its following CAV in the previous lane, which should start accelerating to merge the recently generated gap.
(4) If the received alert is EVA, it is validated using the location validation, nearby vehicles' state validation, and subsequent action validation (Steps 14–16). Here, only the preceding CAVs of the alert sender are affected and they should conduct alert validation and take action accordingly. Here, it needs to check the locations and subsequent actions of the alert sender and its following CAVs to validate the alert.
(5) If the received alert is BSW, it is validated using driving state validation, location validation, nearby vehicles' state validation, and subsequent action validation (Steps 17–20). In a BSW alert, only the preceding CAV in another lane is affected and it should conduct alert validation and take action accordingly. After the alert is validated, the preceding CAV in another lane stops lane changing immediately. Here, the velocity of the preceding and following CAVs of the alert sender should be similar.

We can summarize the above steps to the following. For an EEBL alert, MDS conducts the *subsequent action validation* for the preceding CAV of alert sender i. For an RHC alert, MDS conducts the *subsequent action validation* for the preceding CAV and following CAV of alert sender i. For a BSW alert, MDS conducts the *subsequent action validation* for the preceding CAV and following CAV of alert sender i, the driving state validation of alert sender i. Compared to BSW, a CoL alert needs additional *lane changing validation* of alert sender i. For an EVA alert, MDS conducts the location validation in *nearby vehicles' state validation* and the *subsequent action validation* for the following CAV of alert sender i.

4 PERFORMANCE EVALUATION

In this section, we present the evaluations of MDS. First, we discuss the real-world CAN bus driving data used in the experiments. Then, we present the performance evaluations in three different aspects: driving state prediction, traffic flow considered driving state prediction, and misbehavior prediction.



(a) Velocity (b) Acceleration (c) Brake (d) Steering angle

Fig. 11. Accuracy of MDN.

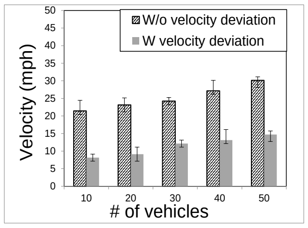
4.1 Driving Data

We built a customized device using the micro-controller (Arduino Duo board and M2 Macchina board) to collect a vehicle's real-time driving data through the CAN bus inside the vehicle at every 0.1 second. We mainly used a Honda Civic 2012 LX vehicle to collect the driving data. The customized device using the M2 Macchina board is able to collect driving data in real time and store the data inside a storage drive. We asked 29 volunteers to drive along four different routes and collected driving data in real time. Each volunteer drove the vehicle around 50 to 60 minutes in a city area where speed limits vary from 25mph to 55mph. The traffic data were collected at different times of the day and, as a result, it contains peak and off-peak traffic conditions (e.g., current traffic flow is less than and equal to the speed limit). Thus, the road condition includes both less and moderate traffic congestion. For each trip, there are different driving maneuvers in terms of speed, deceleration/complete stop, lane changes, and changes of speed limits. For instance, there are 7-11 different lane changing driving maneuvers in total. We also collected data during one snowy afternoon where the traffic congestion was moderate. From the CAN bus, we collected velocity, steering wheel angle, steering wheel momentum, heading direction, statues of turn signal indicators, brake pedal status, gas pedal status, and engine rotation status.

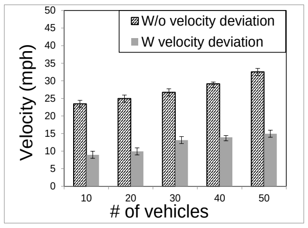
4.2 Driving State Prediction

Here, we used the real driving data to evaluate the MDN model to predict the driving state of vehicles. Inside the MDN model, we used a Long Short Term Memory (LSTM) network as an RNN network (with three hidden layers, three 128-fully connected layers, and one softmax activation function as an output layer for prediction). We set the number of mixtures to 20. We used a variant of LSTM, instead of Gated Recurrent Unit (GRU). We found that LSTM performs better than GRU for our experiments. We tried a different number of hidden layers (3, 6, and 10) and a different number of (128, 256, and 512) fully connected components inside LSTM. From our experiments, we finally chose three 128-fully connected hidden layers inside LSTM. We used the Adam optimizer inside LSTM to optimize the MDN network. We used 1000 epochs at each time to find the convergence of the optimizer. We used a shared GPU server consisting of a Nvidia Tesla K20c GPU with 128GB RAM and 500GB storage. We used the Python programming language in this scenario. We used Edward [1], Keras [2], and TensorFlow [3] API frameworks to implement the MDN model. Edward is a Python library for probabilistic modeling; Keras is a high-level neural networks framework, written in Python and capable of running on top of TensorFlow; TensorFlow is a foundation Python library that can be used to create deep learning models directly or by using wrapper libraries. In general, we used the time window from $(t - 10)$ to $(t - 30)$ to predict the driving state at t. Based on the performance of the prediction model, we finally adjusted the time window as $(t - 10)$.

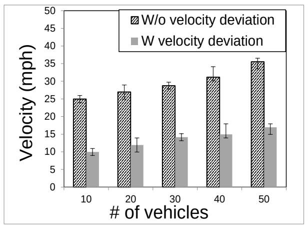
Mean Average Percentage Error (MAPE) represents the prediction accuracy as a percentage (MAPE $= \frac{\sum_{i=1}^{n} \frac{|v_i - \tilde{v_i}|}{v_i}}{n}$, where $\tilde{v_i}$ is the i^{th} estimated velocity, v_i means the i^{th} actual velocity, and n is the number



(a) 35 mph



(b) 45 mph



(c) 55 mph

Fig. 12. Velocity estimations w.r.t. different speed limits.

of velocity values. Fig. 11 shows the accuracy of the MDN model in terms of velocity, acceleration, applied brake, and steering angle. More specifically, Fig. 11a shows the overall velocity prediction accuracy of MDN model for real driving data. We can that than the highest MAPE is around 12%. Next, Fig. 11b shows the MAPE of acceleration predictions for driving data of the 29 persons. We can see that the MDN model causes more acceleration prediction error than in velocity prediction as the range of range of acceleration is lower than the range of velocities. Similarly, Fig. 11c and Fig. 11d shows the MAPEs of brake and steering angle, respectively. From the attack detection accuracy (presented later), we can say that the prediction accuracy of the MDN model is acceptable. The MDN model performs better for velocity, acceleration, and brake status predictions.

4.3 Traffic Flow Considered Driving State Prediction

Recall that we improved the existing Krauss traffic flow model by additionally considering velocity deviations of different vehicles. This experimental study aims to see the effectiveness of our improvement. We used SUMO traffic simulator [15] to simulate the scenario where there are multiple vehicles driving on a single road section. First, we set the number of vehicles and distributed their initial locations. We assumed that the initial velocities of vehicles are equal to the speed limit and use the initial velocities as inputs to Equation (6) to calculate synchronized inter-vehicle distance (D_t). Then, we randomly choose a value in $[0, D_t]$ as the inter-vehicular distance for each pair of vehicles.

Based on the speed limit and inter-vehicle distance as inputs, SUMO calculates the velocities of all vehicles using its own traffic flow model and simulates vehicle movement. We use these velocities of vehicles as the ground truth. Next, we implemented the Krauss traffic flow model inside SUMO (as shown in Figure 10) with and without the consideration of velocity deviation. Krauss traffic flow model also takes initial velocity and inter-vehicle distance as inputs where initial velocity is taken from the trace to use the trace-driven velocity deviations and inter-vehicle distance is calculated as described above. Each simulated vehicle is randomly mapped to a vehicle in the 29 users in the real dataset. Thus, we calculated the velocity deviation for each vehicle using its current and previous 10 seconds' velocities as in Equation (12) to simulate the velocity deviation of the real users. We used the TraCI Python interface inside SUMO to implement the traffic flow. As shown in Figure 10, we calculated the velocities of all vehicles with and without the consideration of velocity deviation, respectively, at every 0.1 second. In this way, each vehicle behaves autonomously and moves with respect to the inter-vehicle distance (D_t), current velocity, and acceleration. We changed the speed limit three times (i.e., 35 mph, 45 mph, and 55 mph) to simulate different driving scenarios. To simulate different traffic flow, we also varied the number of vehicles from 10 to 50 with an increasing 10 at each step.

Fig. 12 shows the velocity prediction error of different numbers of vehicles with/without the consideration of velocity deviations. We can see that comparing to the prediction errors from the Krauss traffic flow model without the consideration of velocity deviation, the Krauss traffic flow model with the consideration of velocity deviation from the MDN model generates lower prediction errors. This result verifies the effectiveness of our proposed method to improve the Krauss traffic flow model. We can also see that when the number of vehicles changes, the prediction errors increase compared with the ground truth as more vehicles cause more positioning error. Also, we can see that when the speed limit increases, the prediction errors increase as it causes higher positioning error.

4.4 Misbehavior Detection

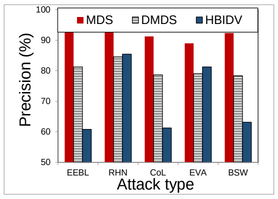
We now evaluate the proposed MDS in misbehavior detection. We used SUMO traffic simulation with 50 vehicles driving in a two-lane 2500-meter long roadway. We fixed the number of attacker vehicles as 15 out of the 50 vehicles. We distributed all vehicles such that each attacker vehicle is in between two non-attacker vehicles so that there are enough non-attacker vehicles to validate the received alert. we considered three different speed limits (i.e., 35 mph, 45 mph, and 55 mph) separately to simulate different traffic flows. Based on our collected real driving dataset for 29 users, we used the methods in Figure 10 to generate each vehicle's velocity over driving time in testing. Each simulated vehicle is randomly mapped to a vehicle in the 29 users. We again used TraCI Python interface inside SUMO to change the velocity of each vehicle in testing. Here, We considered a fixed transmission range of 1000 meters for each vehicle. We simulated the real alert scenario and false alert scenario at alternating 10 seconds for the five different attack scenarios one by one (i.e., EEBL, RHC, CoL, EVA, and BSW). The real alerts are generated by all the vehicles while the false alerts were generated by all the attacker vehicles.

We compared the performances of the proposed MDS with two existing methods in the following.

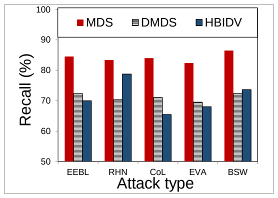
- The Data-centric Misbehavior Detection Scheme (DMDS) [22]. It uses the received message sending time and its estimated message transmission time to estimate the location of the alert sender and compares it with its reported location to check the truthfulness of the alert. Thus, DMDS is suitable for critical scenarios which require quick response time. In DMDS, the alert sender could insert false message sending time to make the calculated location the same as the reported fake location. Also, DMDS cannot estimate location accurately when the change of vehicle velocity (or traffic flow) is significant. In our implementation of DMDS, the alert sender inserts the false message sending time at every five false alert. Assume that there are β vehicles between the alert sender and the alert receiver. When generating the false message sending time, to make it believable, the alert sender makes sure that there are still β vehicles between the alert sender and the alert receiver.
- The Host Based Intrusion Detection in VANETs (HBIDV) [28]. An alert receiver only uses the traffic flow model introduced in Section 2.4 to estimate the alert sender's current location based on its previous location at the time of the reported event. The alert receiver calculates the traffic flow rate of a region centered by the location of the reported event as an input of the traffic flow model. However, the traffic flow rate, without the consideration of velocity deviation of the alert sender, is only able to predict the approximate location of the alert sender (i.e., within a certain road section). Also, HBIDV cannot detect the subtle changes of driving state of an alert sender to validate the alert, so it is not suitable for critical scenarios which require a quick response.

To evaluate the three methods, we measured the precision and recall metrics. *Precision* represents the fraction of truly attacker vehicles identified among all attacker vehicles identified. *Recall* means the fraction of truly attacker vehicles identified over the total number of attacker vehicles.

Figures 13, 14, and 15 show the precision and recall of the three different methods when the number of vehicles is set to 50 and the speed limit is set to 35mph, 45 mph, and 55 mph, respectively. Overall, the performance of the three different methods are as follows: MDS>DMDS>HBIDV. MDS performs better than the other two methods. As discussed in Algorithm 1 in Section 3.3, MDS considers previous driving states and locations of the alert sender with overall traffic flow to check the consistency of reported

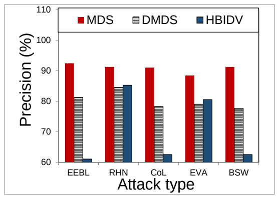


(a) Precision

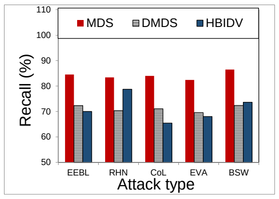


(b) Recall

Fig. 13. Comparison of misbehavior detection methods (35mph).

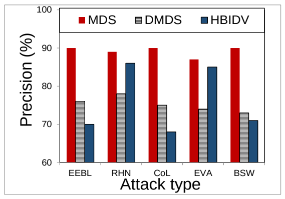


(a) Precision

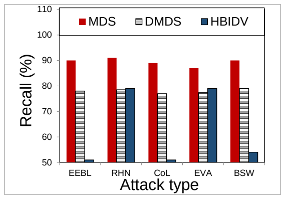


(b) Recall

Fig. 14. Comparison of misbehavior detection methods (45mph).



(a) Precision



(b) Recall

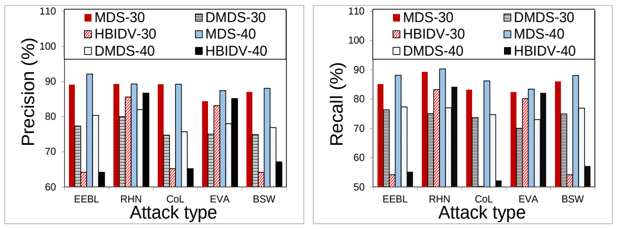
Fig. 15. Comparison of misbehavior detection methods (55mph).

driving state. In addition, it considers the subsequent actions of the alert sender and its nearby CAVs to validate the alert. In DMDS and HBIDV, only location is used to validate the received alert. Further, HBIDV only estimates rough location, and though DMDS can detect subtle location change, the alert sender could insert false message sending time to make the calculated location the same as the reported fake location. In addition, DMDS cannot conduct accurate location estimation when there is a change of vehicle velocity or traffic flow rate. As a result, the precision and recall of MDS are higher than DMDS and HBIDV, and those of DMDS are higher than HBIDV.

DMDS performs better than HBIDV in the EEBL, CoL, and BSW alert scenarios, while performs worse than HBIDV in the RHN and EVA alert scenarios. In the former scenarios, traffic flow rate does not significantly change and a CAV's velocity keeps similar, while in the latter scenarios, the traffic flow rate significantly changes and a CAV's velocity changes greatly. DMDS assumes that a CAV's velocity keeps the same, and HBIDV can take into account the traffic flow rate change by using the traffic flow model for rough vehicle location estimation (but cannot detect subtle location changes). As a result, DMDS can more accurately predict vehicle location than HBIDV in the former scenarios, but less accurately predict vehicle locations than HBIDV in the latter scenarios.

We also see that the performance of MDS slightly decreases, and the performance of DMDS and HBIDV decrease when the speed limit increases. Higher speed limit causes higher positioning error due to GPS positioning error, which affects the accuracy in location estimation. Since MDS considers other factors such as driving states, subsequent behaviors, and nearby vehicles' driving states, it is not significantly influenced by the velocity increase.

We also performed the experiments again by changing the number of vehicles for three comparative methods. Fig. 16 show the precision and recall of three different methods when the number of vehicles was set to 30 and 40, respectively and the speed limit is 55 mph. Here, "Proposed-30" means that the number of vehicles is set to 30 in the MDS, "DMDS-30" means the number of vehicles is set to 30 in the existing DMDS system, "HBIDV-30" means the number of vehicles is set to 30 in the existing HBIDV



(a) Precision (b) Recall

Fig. 16. Comparison of misbehavior detection with a variable number of vehicles.

system, and so on. We can see that the performances of all methods follow: MDS>DMDS>HBIDV. Due to the same reasons described above, the proposed MDS system outperforms other methods as it can accurately detect the false alert in different scenarios. DMDS performs better than HBIDV in EEBL, CoL, and BSW alert scenarios while performs worse than HBIDV in the RHN and EVA alert scenarios due to the same reasons explained before.

5 RELATED WORK

Misbehavior detection. There are several methods proposed for misbehavior detection in vehicular ad-hoc networks [6, 8, 9, 12, 20–22, 28]. In one set of works [8, 12, 20], the misbehavior detection methods mostly use the currently reported location along with the cooperation between different vehicles to evaluate the consistency between previously received BSMs and currently received BSMs. In another group of works [21, 22], the location of the sender vehicle is estimated using the message sending time in the received message and message transmission time and then, the reported location is compared with the estimated location. The work in [21] additionally considers the velocity difference of two nearby vehicles. If the velocity difference is negligible, then the received alert is true. If not, then one of the two vehicles is malicious. Some of the other existing misbehavior detection methods [6, 9, 28] use different traffic flow models to estimate the current location of the alert sender from its previous location and then validate the alert scenario. These works consider that the received alert is valid if the estimated location and received location are within a certain region (i.e., a road section). However, these methods can only estimate a vehicle's location approximately and they cannot be applied to critical scenarios that require to estimate location accurately and to check the driving state of an alert sender vehicle as they do not consider the velocity deviation with the driving state of the alert sender vehicle. In addition, the change of locations in some scenarios are subtle and these methods cannot validate the alert in these scenarios. In this paper, we propose a misbehavior detection system to accurately predict an alert sender's current driving state (including location) based on its previous driving states, its subsequent actions, and overall traffic flow to identify the false information from the malicious vehicles.

Driving state predictions. There is a large set of existing works [19, 25, 26] that predict the driving state using the model predictive control techniques. Miyajima et al. [19] identified different driving states (e.g., car-following and pedal operation patterns) based on a GMM using the relationship between the inter-vehicular distance and velocity. Lefevre et al. [17] designed a longitudinal predictive controller of an autonomous vehicle to mimic the car following behaviors and they considered the possible accelerations from a human driver in the same situations. The authors applied a set of comfort and safety constraints in the controller before applying the human acceleration. Wang et al. [25] presented an automatic car following behavior to avoid the possible collisions and to activate automatic brake based on the data collected for the desired throttle pressure and brake pressure. Xu et al. [26], a driving style-oriented driver modeling is presented to control vehicle velocity using a neural network-based controller. To tackle the divergence and local mutability of real-world vehicle data, the neural network is designed and trained to mimic human behaviors on applying gas or brake pedals. Some other works [4, 11] use smartphone sensors (i.e., accelerometer, gyroscope, and magnetometer) to obtain location, speed, acceleration, deceleration, and steering angle to analyze the different driving maneuvers (e.g., speeding up/slowing down). In our work, we predict different driving states using an MDN model incorporating RNN which is able to consider the periodicity of driving states in different traffic scenarios for high prediction accuracy.

6 CONCLUSION

Previously proposed false information detection methods for V2V communication only estimate the alert sender's location and are not effective in critical scenarios that require to take actions in real time. In this paper, we presented the data-driven MDS for the CAV systems, which considers both location and driving states of an alert sender and its nearby vehicles to effectively detect false alerts in such critical scenarios. *First*, MDS predicts the driving state of an alert sender based on a GMM-based MDN model. Inside the MDN model, MDN uses RNN as a deep learning network to consider the previous driving states to predict the current driving state. *Second*, MDS incorporates velocity deviation prediction of an alert sender from MDN network into the existing Krauss traffic flow model. In this way, it calculates the velocity of an alert sender based the inter-vehicular distances and velocities of its nearby CAVs. *Finally*, MDS considers driving states and locations of an alert sender and its nearby CAVs together to validate the received alerts from CAVs. In addition, to reduce the false positive rate, MDS also considers the subsequent actions of an alert sender and its nearby CAVs for a short period of time (e.g., 0.3 seconds). To validate MDS, we performed extensive simulation studies based on a real driving dataset collected from 29 participants and the SUMO traffic simulator. The comparative studies with the two other existing methods show that MDS has higher detection accuracy rate (i.e., higher precision and recall). In the future, we will consider the personalized driving behaviors with the overall traffic flow to validate different alerts.

ACKNOWLEDGMENTS

This research was supported in part by U.S. NSF grants CCF-1822965, OAC-1724845, ACI-1719397, and CNS-1733596 and Microsoft Research Faculty Fellowship 8300751. The authors are grateful to the anonymous reviewers for their valuable suggestions. The authors would like to thank Mr. Liuwang Kang for his valuable discussions and comments.

REFERENCES

[1] [n. d.]. Edward: A library for probabilistic modeling, inference, and criticism. http://edwardlib.org/, note = Accessed: 2018-04-01.
[2] [n. d.]. Keras. https://keras.io/. Accessed: 2018-04-01.
[3] [n. d.]. TensorFlow. https://www.tensorflow.org/. Accessed: 2018-04-01.
[4] Alberto Diaz Alvarez, Francisco Serradilla Garcia, José Eugenio Naranjo, Jose Javier Anaya, and Felipe Jimenez. 2014. Modeling the driving behavior of electric vehicles using smartphones and neural networks. *IEEE ITSM* 6, 3 (2014).
[5] Fan Bai, Tamer Elbatt, Gavin Hollan, Hariharan Krishnan, and Varsha Sadekar. 2006. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proc. of AutoNet*.
[6] Rajesh P Barnwal and Soumya K Ghosh. 2012. Heartbeat message based misbehavior detection scheme for vehicular ad-hoc networks. In *Proc. of ICCVE*.
[7] Christopher M Bishop. 1994. Mixture density networks. (1994).
[8] Norbert Bißmeyer, Christian Stresing, and Kpatcha M Bayarou. 2010. Intrusion detection in VANETs through verification of vehicle movement data. In *Proc. of VNC*.
[9] Tarek Bouali, Sidi-Mohammed Senouci, and Hichem Sedjelmaci. 2016. A distributed detection and prevention scheme from malicious nodes in vehicular networks. *Internation Journal of CS* 29, 10 (2016).
[10] Joan Fisher Box. 1987. Guinness, Gosset, Fisher, and small samples. *Statistical science* (1987).

[11] Haluk Eren, Semiha Makinist, Erhan Akin, and Alper Yilmaz. 2012. Estimating driving behavior by a smartphone. In *Proc. of IV*.

[12] Mainak Ghosh, Anitha Varghese, Arzad A Kherani, and Arobinda Gupta. 2009. Distributed misbehavior detection in VANETs. In *Proc. of WCNC*.

[13] Renate Haeuslschmid, Bastian Pfleging, and Florian Alt. 2016. A design space to support the development of windshield applications for the car. In *Proc. of CHI*.

[14] John Harding, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. 2014. *Vehicle-to-vehicle communications: Readiness of V2V technology for application*. Technical Report.

[15] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On ASM* (2012).

[16] Stefan Krauß, Peter Wagner, and Christian Gawron. 1997. Metastable states in a microscopic model of traffic flow. *PR E* 55, 5 (1997).

[17] Stéphanie Lefèvre, Ashwin Carvalho, and Francesco Borrelli. 2016. A learning-based framework for velocity control in autonomous driving. *IEEE Trans. on ASE* 13, 1 (2016).

[18] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of ISCA*.

[19] Chiyomi Miyajima, Yoshihiro Nishiwaki, Koji Ozawa, Toshihiro Wakita, Katsunobu Itou, Kazuya Takeda, and Fumitada Itakura. 2007. Driver modeling based on driving behavior and its evaluation in driver identification. *Proc. IEEE* 95, 2 (2007).

[20] Maxim Raya, Panagiotis Papadimitratos, Imad Aad, Daniel Jungels, and Jean-Pierre Hubaux. 2007. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on SAC* 25, 8 (2007).

[21] Ziwei Ren, Wenfan Li, and Qing Yang. 2009. Location verification for vanets routing. In *Proc. of WIMOB*.

[22] Sushmita Ruj, Marcos A Cavenaghi, Zhen Huang, Amiya Nayak, and Ivan Stojmenovic. 2011. On data-centric misbehavior detection in VANETs. In *Proc. of VTC*.

[23] Ankur Sarker, Chenxi Qiu, and Haiying Shen. 2016. A Decentralized Network with Fast and Lightweight Autonomous Channel Selection in Vehicle Platoons for Collision Avoidance. In *Proc. of MASS*.

[24] Ankur Sarker, Chenxi Qiu, and Haiying Shen. 2017. Quick and Autonomous Platoon Maintenance in Vehicle Dynamics For Distributed Vehicle Platoon Networks. In *Proc. of IoTDI*.

[25] Jianqiang Wang, Lei Zhang, Dezhao Zhang, and Keqiang Li. 2013. An adaptive longitudinal driving assistance system based on driver characteristics. *IEEE Trans. on ITS* 14, 1 (2013).

[26] Li Xu, Jie Hu, Hong Jiang, and Wuqiang Meng. 2015. Establishing style-oriented driver models by imitating human driving behaviors. *IEEE Trans. on ITS* 16, 5 (2015).

[27] Li Yan, Haiying Shen, and Kang Chen. 2015. TSearch: Target-Oriented Low-Delay Node Searching in DTNs With Social Network Properties. In *Proc. of INFOCOM*.

[28] Kamran Zaidi, Milos B Milojevic, Veselin Rakocevic, Arumugam Nallanathan, and Muttukrishnan Rajarajan. 2016. Host-based intrusion detection for VANETs: a statistical approach to rogue node detection. *IEEE Trans. on VT* 65, 8 (2016).

[29] Heiga Zen and Andrew Senior. 2014. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proc. of ICASSP*.

Received August 2018; revised October 2018; accepted October 2018