Road Context-aware Intrusion Detection System for Autonomous Cars

Jingxuan Jiang[1*], Chundong Wang[2†],
Sudipta Chattopadhyay[2], and Wei Zhang[1]

[1] School of Control Science and Engineering, Shandong University, China
jingxuan_jiang@mail.sdu.edu.cn, davidzhang@sdu.edu.cn
[2] Singapore University of Technology and Design, Singapore
cd_wang@outlook.com, sudipta_chattopadhyay@sutd.edu.sg

Abstract. Security is of primary importance to vehicles. The viability of performing remote intrusions to the in-vehicle network has been manifested. For unmanned autonomous cars, limited work has been done to detect such intrusions, while existing intrusion detection systems (IDSs) embrace limitations against strong adversaries. We hence consider the very nature of autonomous car and leverage the *road context* to design a novel IDS, named *R*oad context-*a*ware *IDS* (RAIDS). Given an autonomous car driving along continuous roads, road contexts and genuine frames transmitted on the car's in-vehicle network should resemble a regular and intelligible pattern. RAIDS employs a lightweight machine learning model to extract road contexts from sensory information (e.g., camera images and sensor values) used to control the car. With the road context, RAIDS validates corresponding frames observed on the in-vehicle network. Anomalous frames that substantially deviate from road context will be discerned as intrusions. We have built a prototype of RAIDS with neural networks, and done experiments on a Raspberry Pi with extensive datasets and meaningful intrusion cases. Evaluations show that RAIDS significantly outperforms state-of-the-art IDS without any road context by up to 99.9% accuracy and short response time.

Keywords: Autonomous Car · Road Context · Intrusion Detection

1 Introduction

Security is critical for vehicles. A modern automobile embodies a protocol, like the prevalent Control Area Network (CAN) [11], for in-vehicle communications among its electrical subsystems, such as the steering wheel, brake, and engine, each of which is monitored and controlled through an electronic control unit (ECU). Researchers managed to manifest concrete intrusions to ECUs of manned vehicle to cause a breakdown or traffic accident [3, 6, 12, 25]. Today, many

*This work was done when J. Jiang was an intern at Singapore University of Technology and Design.

†C. Wang is the corresponding author (cd_wang@outlook.com).

practitioners and researchers are developing self-driving autonomous cars, which, undoubtedly, demand particular care for security and safety [14, 17]. However, limited work has been done on designing an intrusion detection system (IDS) for the in-vehicle network of autonomous car. Existing IDSs even have limitations against strong adversaries. Take the state-of-the-art CIDS [6] for example. In accordance with its knowledge of all existing ECUs, CIDS tracks down anomalies when an original ECU stops sending frames or an ECU belonging to adversaries injects frames. Nevertheless, CIDS should be oblivious of a compromised ECU sending forged frames. If a strong adversary can manipulate an original ECU to deliver fake frames, CIDS would malfunction as the fingerprint of the ECU is not peculiar. As a result, such an attack model is beyond the capability of CIDS.

There is a fact that has not been considered in designing IDS to protect in-vehicle network: all frames transmitted on the CAN bus are generated due to the decisions made by the vehicle driver and it is the *road context* that guides a driver to make those decisions. Human drivers have highly individualized experiences and habits, and react differently to the same road context, like a stop sign or a road bend. It is hence impractical to design an IDS with road context for manned vehicles. By contrast, an autonomous car is orthogonal to manned vehicles concerning the very nature of 'driver'. In an autonomous car, decisions are made by a well-trained self-driving model upon dynamic road contexts obtained through multiple sensors [2, 4]. Therefore, the road context and corresponding control signals, which eventually result in frames transmitted on the CAN bus, shall resemble a regular and intelligible pattern. Given an intrusion with forged frames upon continuous road contexts, a violation of the pattern can be perceivable.

Motivated by this observation, we develop a holistic IDS, i.e., ***R***oad context-***a***ware **IDS** (**RAIDS**), for autonomous cars to detect anomalous CAN frames forged by strong adversaries. Main ideas of RAIDS are summarized as follows.

- RAIDS is a two-stage framework that mainly consists of two neural networks to extract road context from sensory information (e.g., images taken by cameras, distances to front objects, etc.) and validate the genuineness of CAN frames, respectively, for the purpose of intrusion detection. Both neural networks are designed to be lightweight and efficient regarding the computational resources of an in-vehicle embedded computing system.
- To extract road contexts, the neural network at the first stage of RAIDS processes camera images and other sensory information that are concurrently used by the self-driving model to control the car. The second stage of RAIDS is a binary classifier that verifies whether the frames observed on the CAN bus are abnormal or not with regard to the extracted road contexts.

We have built a prototype of RAIDS[‡]. A convolutional neural network (CNN) makes the backbone of RAIDS's first stage for extracting and abstracting road contexts from camera images. The second stage of RAIDS mainly leverages linear layers to efficiently discern anomalous CAN frames with extracted road context. To evaluate RAIDS, we follow state-of-the-art work [21] and implement an IDS that learns from historical CAN frames without road context. We run both IDSs

[‡]The source code of RAIDS is available at `https://github.com/cd-wang/RAIDS`.

in a Raspberry Pi with extensive datasets. On defending two types of intrusions, i.e., abrupt and directed intrusions, RAIDS substantially outperforms the IDS without road context by up to 99.9% accuracy and short response time.

The rest of this paper is as follows. In Section 2, we present the background and related works of RAIDS. In Section 3, we show the motivation and attack model for RAIDS. In Section 4, we detail the design of RAIDS. In Section 5, we present the evaluation results of RAIDS, and conclude the paper in Section 6.

2 Background & Related Works

IDS Multiple IDSs have been proposed targeting the in-vehicle network [6, 11, 13, 15, 19]. An automobile is made of multiple electrical subsystems, each of which has an ECU to communicate with other subsystems to control the vehicle. ECUs encapsulate data in frames and put them on the CAN bus. A CAN frame contains no identity information of sender or receiver for simplicity. The lack of identity in CAN frames facilitates adversaries in fabricating hazardous messages. Worse, modern vehicles are being connected to the outside world via multiple channels, which leave exploitable attack vectors for adversaries to leverage.

Many IDSs analyze normal CAN frames to detect anomalous ones. Müter and Asaj [15] found that CAN frames are more 'regular' than frames found on computer networks, which leads to a relatively low *entropy* for CAN frames. Hence injecting or dropping CAN frames should increase the entropy of in-vehicle network and in turn expose an intrusion. Song et al. [19] worked in a similar fashion but used the time interval between CAN frames to inspect suspicious frames. Taylor et al. [21] emphasized on the data carried in CAN frames and proposed a recurrent neural network (RNN)-based anomaly detector. Their RNN is trained with historical normal CAN frames to predict forthcoming frames and apprehend abnormal ones. Nonetheless, their experiments for detecting anomalous frames were done by manually flipping unused bits of data in a CAN frame to emulate an 'unusual case'. Such a manipulation is irrational as skilled adversaries must have a good knowledge of transmitted data and tend to fabricate meaningful but harmful frames. Wasicek et al. [25] proposed to learn the 'intra-vehicle context' by collecting the values of multiple sensors installed in a vehicle's subsystems and building reference models to detect anomalies. Note that their 'context' is the internal context inside a vehicle, not road context. Meanwhile, Cho and Shin [6] proposed Clock-based IDS (CIDS) that used the clock skew of ECUs to fingerprint them. Leveraging the unique clow skew of each ECU, CIDS can not only detect the occurrence of intrusions, but also locate the compromised ECU.

Autonomous Car & Neural Network We consider an autonomous car that is computer-controlled most of the time except for emergency cases, such as an intrusion to in-vehicle network. Multiple sensors are installed to control an autonomous car, including cameras, ultrasonic distance sensors, radar, etc. Such sensory information reflects and resembles real-world road context, and advises the self-driving model to generate control signals. Control signals are transformed to data encapsulated in frames transmitted on the CAN bus.

Numerical sensor values, like the distance to front objects, are computer-readable and can be directly utilized by the self-driving model of autonomous car. The camera images, however, must be processed to acquire high-level informative properties. Nowadays, neural networks have emerged as the mainstream approach that deals with images for self-driving. For example, the convolutional neural network (CNN) has been proved to be effective in extracting image features to maneuver the autonomous car [2, 10]. A CNN makes use of convolutional layers that apply multiple kernels to extract embedded visual features. A kernel is a small matrix of numbers. An image can be viewed as a large matrix that comprises many small sub-matrices with the kernel size. Convolutional layer *convolves* each kernel over sub-matrices to do matrix multiplication. The output of a convolutional layer is thus a feature map that bundles results of convolving multiple kernels. In a CNN, feature maps of several convolutional layers, after being computed through hidden layers for reduction of computations and avoidance of overfitting, will eventually make a vector that resembles the features per image. Such a feature vector is expressive and meaningful in image understanding [4].

3 Problem Formulation

Motivation Most IDSs were designed regarding human-driven vehicles. Many technology giants, startups, and academic researchers are developing autonomous cars, which, undoubtedly, demand particular care for security and safety.

Limited work has been done on detecting intrusions to the in-vehicle network of autonomous car. Worse, state-of-the-art IDSs embrace limitations against strong adversaries. For example, CIDS [6] is able to detect intrusions when a foreign ECU injects messages or an existing ECU stops sending messages based on its knowledge of fingerprints (i.e., clock skews) of ECUs. However, if adversaries compromise an ECU to send fake messages, CIDS will be ineffective as the fingerprint is not suspicious. CIDS should be oblivious of compromised ECUs sending fake messages. Such intrusion cases are beyond the capability of CIDS.

In practical, CAN frames are generated when drivers encounter dynamic road contexts. Assume that a stop sign is ahead. A driver must decelerate and then stop the car for a moment. The ECU of accelerator accordingly produces CAN frames with decreasing speed values. Fig. 1(a) and Fig. 1(b) illustrate how two human drivers react when they move towards a stop sign. One driver gradually reduces speed. The other one does so only when being close to the stop sign. Because human drivers have different experiences and habits, they have different reactions that entail distinct CAN frames.

When a car is controlled by a well-trained self-driving model, its behaviors should be smooth and stable. As shown by Fig. 1(c), an autonomous car shall start to reduce speed on spotting a stop sign and steadily slow down in order to approach the stop line. This results in CAN frames with consistently decreasing speed values. Concretely, the road context (i.e., a stop sign) and CAN frames (i.e., decreasing speed values) construct a regular and consistent pattern for autonomous car. Assume that adversaries compromise the accelerator

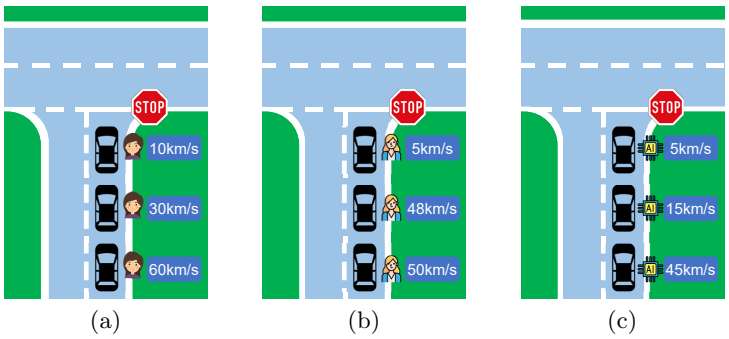


Fig. 1. An Illustration of Reactions of Human-driven and Autonomous Cars on a Stop Sign: (a) Human-driven Car 1; (b) Human-driven Car 2, and (3) Autonomous Car

of autonomous car and continually put CAN frames with non-decreasing speed values. These abnormal frames are easy to be ruled out as they significantly deviate from the pattern supposed for a stop sign.

To sum up, given a specific road context, the consequential CAN frames generated by an autonomous car are regular and predictable. If we monitor ongoing road context and validate against observed CAN frames, anomalous frames shall be detectable. This observation motivates us to design a new IDS.

Attack Model Several network connections exist in an automobile to help it communicate with the outside world. These connections yet provide attack vectors for adversaries to exploit. We assume that strong adversaries further have good knowledge of in-vehicle network, including the format and frequency of CAN frames issued by an ECU, and also manage to force an ECU to encapsulate and send their data in CAN frames. With such knowledge, adversaries are able to remotely access and manipulate critical ECUs of an autonomous car, such as the steering wheel, brake, and accelerator. In this paper, we consider an attack model that is beyond the capability of state-of-the-art IDSs, i.e., *forgery attack*.

The process of a forgery attack is as follows. Once adversaries compromise an ECU, they first intercept the normal frames sent and received by the ECU to study the ECU's behavior and data format. Then adversaries start forging and sending CAN frames strictly with the original frequency. The data put in forged frames is yet made either inappropriate or opposite due to the malicious intentions of adversaries. For example, upon a left turn, adversaries may replace CAN frames of the steering wheel with right turn angles so as to wreck the car.

Forgery attack has two variants.

- *Abrupt intrusion*: adversaries abruptly place anomalous CAN frames with abnormal data at a random time to cause a disorder.
- *Directed intrusion*: adversaries monitor the road context at runtime and, upon a specific scenario, like a road bend or traffic light, place anomalous CAN frames that significantly violate the road context.

Assuming the autonomous car is unaware of any intrusion, the impact of directed intrusion is more detrimental, as the CAN frames it imposes shall inflict a sudden flip to the vehicle's state, like the aforementioned right turn upon a left turn.

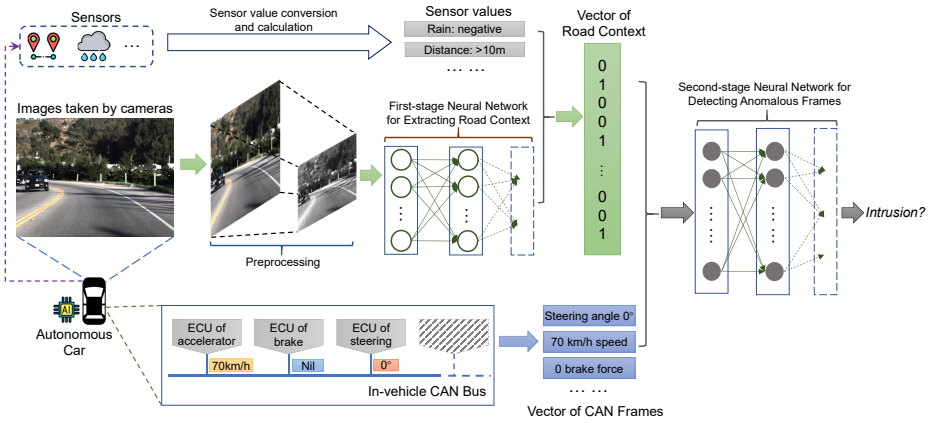


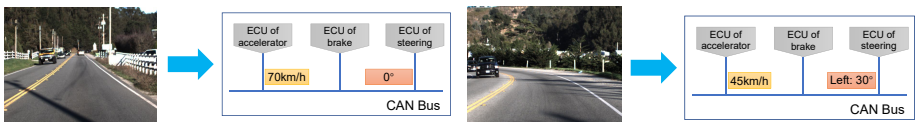
Fig. 2. An Illustration of RAIDS's Architecture

4 RAIDS

4.1 Overview of RAIDS

The essence of RAIDS is to leverage the ongoing road context to validate whether CAN frames on the in-vehicle network are normal or not for an autonomous car. If CAN frames closely match the corresponding road context, RAIDS deems that there is no security threat. Otherwise, RAIDS will report an intrusion.

Fig. 2 illustrates the architecture of RAIDS. As shown by the leftmost of Fig. 2, the ongoing road context is reflected by a variety of sensory information, such as the distances to surrendering objects and camera images showing the front scene. Numerical sensory information is computer-readable while camera images must be processed. The self-driving model depends on sensory information to decide how to maneuver the autonomous car. Such sensory information is also delivered to RAIDS. RAIDS is mainly composed of two neural networks. One neural network is responsible for processing the camera images which cannot be instantly utilized. The image is first preprocessed through techniques like normalization and centering. Then RAIDS uses one neural network, as shown at the central part of Fig. 2, to extract and abstract image features. These image features will be concatenated with other numerical sensory information to make a vector of road context. On the other hand, as illustrated by the lower half of Fig. 2, the self-driving model produces control signals upon the sensory information, which eventually conveys a number of CAN frames transmitted on the in-vehicle network. These CAN frames are formulated into another vector that is fed along with the vector of road context as two inputs to the second neural network of RAIDS. As shown by the rightmost of Fig. 2, with well-trained parameters learned from historical road contexts and CAN frames, the second neural network shall tell whether abnormal frames emerge on the CAN bus or not. RAIDS immediately informs the self-driving model once an anomaly is detected.



(a) The Road Context of a Straight Road (b) The Road Context of a Road Bend

Fig. 3. An Illustration of the Impact of Road Context on CAN Frames

If an intrusion is reported, RAIDS suggests that the self-driving model should 1) first disable external network connections to block remote adversaries, 2) stop the vehicle for emergency if possible, and 3) raise a switch request to human driving. These steps aim to mitigate the impact of intrusions.

4.2 Road Context

We define the road context as *the information an autonomous car is encountering when it is cruising*. In summary, the road context includes but is not limited to, 1) road conditions, like traffic lights, the bend, joint, and fork of roads, 2) pedestrians, vehicles, obstacles, and bumps around the car, 3) weather conditions, like the rain, fog, and snow, and 4) the sunrise, sunset, and tunnel lights. These road contexts are perceived by sensors installed in the car, including cameras, ultrasonic distance sensors, water sensor, etc.

The road context determines control signals issued by the self-driving model. Different road contexts entail different signals, which in turn generate different CAN frames. Fig. 3 instantiates the impact of road curves on the control signal. As shown in Fig. 3(a), on a highway that is straight, the self-driving model demands the autonomous car to move straightforward and run at a velocity of 70km/h. By contrast, upon a road bend as illustrated in Fig. 3(b), the framework shall decrease the car's velocity to 45km/h and turn to the left with an angle of 30°. Assuming that on the road shown in Fig. 3(b), a frame with a steering angle of 0° for moving straightforward, rather than the rational frame of left turn with 30°, emerges on the CAN bus, an intrusion should have taken place because the anomalous frame is not congruent with the ongoing road context.

CAN frames are ever-changing due to dynamic road contexts from time to time. The tight relation between road context and CAN frames indicates that road context must be exploited as a crucial parameter to detect intruions initiated onto the in-vehicle network. We hypothesize that the self-driving model of autonomous car is intact and always makes wise and regular decisions upon dynamic road contexts. In other words, an autonomous car is a contrast to manned vehicles in which human drivers may behave inconsistently from time to time even regarding the same road context. In addition, we note that the focus of this paper is on detecting security threats imposed by adversaries onto ECUs and in-vehicle network of a self-driving automobile. Readers may refer to other studies for the vulnerability exploration of deep learning models that drive a vehicle [20, 24] and intrusion detections for the traffic systems [8].

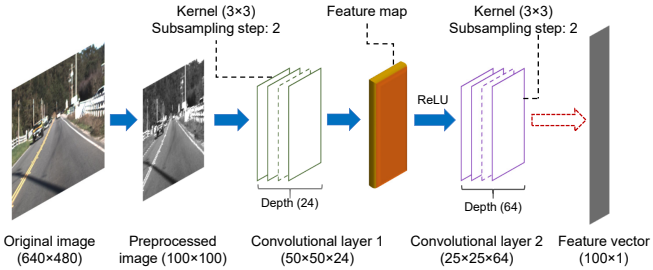


Fig. 4. The Architecture of RAIDS's CNN

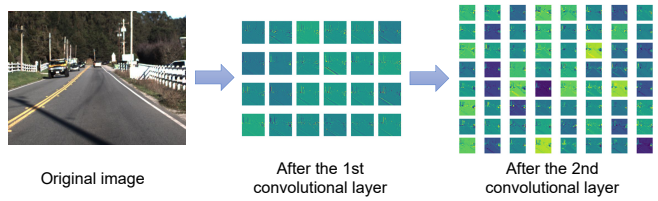
4.3 Extracting Road Context

We first extract the road context to use it for intrusion detection. As a matter of fact, most of the road context is reflected by the front scene that autonomous car is facing, including the aforementioned road condition, traffic light and weather. Such a scene is tracked by multiple sensors. Numerical sensory information, like the distance to front objects, can be directly utilized by RAIDS since they are both human- and computer-readable. The images captured by cameras, however, need to be converted into a format that RAIDS can deploy. As a result, the difficulty of obtaining road context lies in how to process camera images.

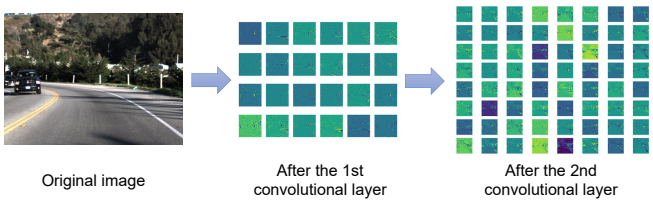
As mentioned in Section 2, the feature vector obtained in a deep neural network is a promising abstraction of road context contained in an image for RAIDS to leverage for intrusion detection. Concretely, we construct a CNN as the backbone of the first stage of RAIDS to extract and abstract the road context from camera images. Fig. 4 sketches the architecture of the CNN employed by RAIDS to process images. It mainly consists of two convolutional layers. After being preprocessed, the first convolutional layer would apply 24 kernels, each of which is 3×3 with a subsampling step[§] of 2, to generate a feature map (50×50×24). This feature map goes through a rectified linear unit (ReLU), which is the activation function used in our implementation, and then reaches the second convolutional layer. The second convolutional layer applies 64 kernels, each of which also has 3×3 size with a subsampling step of 2. The second feature map is hence 25×25×64, and would entail a feature vector of 100×1 after passing one dropout layer and two dense layers.

Fig. 5 exemplifies the feature maps visualized after two convolutional layers for two images from Udacity dataset [23] when they are being processed by the CNN of RAIDS. A comparison between Fig. 5(a) and Fig. 5(b) confirms that different road contexts lead to different intermediate features. In the end, the feature vector of each image would be assembled with numerical sensor values into a new vector as one of the inputs to the second stage of RAIDS.

[§]A subsampling of 2 means that the convolutional layer moves each kernel by 2 , rather than 1, when sliding over the large matrix of image, to reduce the dimensionality of feature map but without losing important information of the image. So the size of feature map for one kernel is $\frac{100}{2} \times \frac{100}{2} = 50 \times 50$.



(a) Straight Road



(b) Road Bend

Fig. 5. An Illustration of Extracting Features from Images by RAIDS's CNN

The CNN extracting image features for RAIDS works synchronously with the self-driving model of autonomous car, because RAIDS should verify CAN frames produced by the self-driving model on the same road context. RAIDS's CNN is simpler than the self-driving model's. The reason is twofold. First, the self-driving model does not terminate with image features but has to accordingly do further computations to determine control signals subsuming left or right steering with a degree, acceleration with a velocity, brake with a force, etc. Second, the feature vector generated by the CNN of RAIDS can be coarse-grained as long as they are sufficiently accurate for detecting intrusions at the second stage of RAIDS.

4.4 Intrusion Detection with Road Context

After obtaining the vector of road context and CAN frames corresponding to the road context, we can establish a model between them by learning over historical records of road contexts and CAN frames. With the model, RAIDS validates whether observed CAN frames approximately match the newly-arrived image and sensor values. A substantial discrepancy would lead to a report of intrusion.

As historical records of sensory information and CAN frames are known as normals, how to detect intrusions on the in-vehicle network regarding road context turns to be a problem of developing a supervised learning model to check CAN frames upon forthcoming sensory information. Assume that N items of sensory information are used for training. The ith sensory information ($0 \leq i < N$) has a vector of road context r_i obtained from the first stage of RAIDS. Still for the ith sensory information, CAN frames issued by ECUs like steering wheel, accelerator, and brake have been collected and presented in a vector c_i. In the perspective of supervised learning, we can make a label λ_i ('1' for normal and '0' for anomaly) for r_i and c_i. Given these N tuples forming a training dataset,

$$\{\langle r_0, c_0, \lambda_0\rangle, \langle r_1, c_1, \lambda_1\rangle, ..., \langle r_i, c_i, \lambda_i\rangle, ..., \langle r_{N-1}, c_{N-1}, \lambda_{N-1}\rangle\}, \quad (1)$$

RAIDS's supervised learning attempts to seek out a model,

$$g(r_i, c_i) \approx \lambda_i, (0 \leq i < N). \tag{2}$$

Or put in another way,

$$g : \boldsymbol{R} \times \boldsymbol{C} \rightarrow \boldsymbol{\Lambda}, \tag{3}$$

in which $\boldsymbol{R}$, $\boldsymbol{C}$, and $\boldsymbol{\Lambda}$ are the domain spaces of road contexts, CAN frames, and labels, respectively.

The function g should be one that the best describes the relationship between $\boldsymbol{R}$, $\boldsymbol{C}$, and $\boldsymbol{\Lambda}$. For a new element of $\boldsymbol{R} \times \boldsymbol{C}$, which will be forthcoming road context r_x and its corresponding vector of CAN frames v_x, RAIDS computes $g(r_x, v_x)$ and obtains a label λ_x. If λ_x is '0', RAIDS deems there would be no intrusion at that moment. Otherwise, RAIDS informs the self-driving model of a possible intrusion on the CAN bus. Consequently, the second stage of RAIDS is formatted as a problem of binary classification. In the prototype of RAIDS, we build a classifier that is mainly composed of two linear layers. There are two reasons to do so. First, r_i, v_i, and λ_i are numerical vectors. Linear layers are sufficient to speculate their relationship. Second, linear layers bring about relatively simpler computations, which are especially efficient concerning the response time of IDS and the computational resources of an embedded computing system.

4.5 Training and Testing

We have built the first and second stages of RAIDS[¶] with a CNN and a binary classifier, respectively, based on Keras [7] and PyTorch [16] frameworks. We follow an end-to-end learning fashion [2] to train RAIDS. The loss function is BCELoss (Binary Cross-Entropy loss) provided by PyTorch for binary classification [18]. We would use six datasets to evaluate RAIDS (more details can be found in Section 5). In each dataset, we use 70% images and CAN frames for training while the 30% remainders are used for the purpose of testing. We note that because datasets have different image sizes, RAIDS would have different implementation variants to deal with respective datasets.

5 Evaluation

We have performance evaluations to answer three questions.

Q1. Does RAIDS achieve high accuracy in intrusion detection? Is the performance of RAIDS stable over different datasets?

Q2. How is the efficacy of RAIDS? Does it cost reasonable response time to detect an intrusion in an embedded computing system?

Q3. Is RAIDS effective in detecting intrusions under more difficult road contexts. e.g., nighttime road conditions?

[¶]The source code of RAIDS is available at `https://github.com/cd-wang/RAIDS`.

Table 1. The Datasets and Intrusions Used to Evaluate RAIDS

Datasets	Sources	Genuine Steering Angle Ranges in Radian	Manipulations of Abrupt Intrusion	Manipulations of Directed Intrusion
Udacity	Udacity self-driving challenge [23]	[-2.05, 1.90]	Randomly select 30% images and for an image, add or subtract a random value in [0.1, 0.9] to its corresponding angle.	Select the largest 15% and smallest 15% angles. Flip the sign of a selected angle if its absolute value is larger than 0.3. If not, add or subtract a random value in [0.5, 1].
Udacity_sim	Udacity simulator [22]	[-0.94, 1.00]		
Apollo	Road Hackers platform in Baidu Apollo Project [1]	[-0.38, 0.21]	Randomly select 30% images and for an image, add or subtract a random value in [0.08, 0.5]to its corresponding angle.	
Chen_2017	Recorded by Sully Chen in 2017 and	[-1.99, 0.55]	Randomly select 30% images and for an image, add or subtract random value	
Chen_2018	2018, respective [5]	[-2.01, 0.68]	in [0.2, 0.9] to its corresponding angle.	
Comma.ai	Comma.ai highway driving [9]	[-1.64, 1.29]	Randomly select 30% images and for an image, add or subtract a random value in [0.25, 1] to its corresponding angle.	

5.1 Evaluation Setup

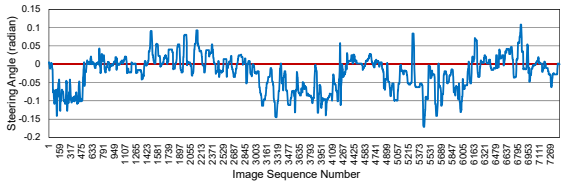
Datasets We have used six datasets from five sources. Their descriptions are presented in Table 1. Except Udacity_sim with images recorded in a synthesized simulator, all other datasets were collected in the real world. They all contain a large number of records with images and data conveyed in CAN frames.

Road Context We place emphasis on the road conditions reflected by camera images, such as lane lines, road bends, and turns. There are two reasons to do so. First, not all datasets provide numerical sensor values. Second, less sensory information imposes more challenges in precisely obtaining road context.

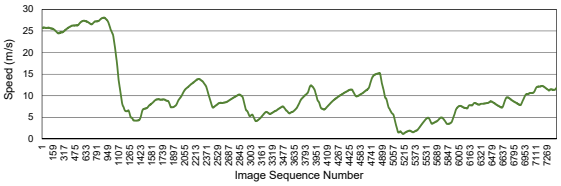
At the standpoint of adversaries, we would focus on intrusions onto the steering wheel. The reason is threefold. First, the steering angle is one vital control signal for autonomous car and attracts wide attention for research. Second, the steering angle is ever-changing along the road while the control signals from accelerator and brake remain relatively stable for a moving vehicle. Fig. 6 sketches two curves for the steering angle and vehicle speed at runtime, respectively, with one Udacity sub-dataset (HMB_6). It is evident that the curve of vehicle speed is much smoother than that of steering angle. Therefore, an intrusion to compromise steering angle is more difficult to be detected. Third, some datasets, like Chen_2017 and Chen_2018, only include the runtime values of steering angle.

Intrusions We consider forgery attacks in evaluation. We have performed abrupt and directed intrusions to the steering angles with each dataset. In contrast to existing works that encapsulated meaningless data in CAN frames or fabricated artificial CAN frames, our intrusions generate a CAN frame with an allowable value that yet does not match the ongoing road context. The rightmost two columns of Table 1 brief how we manipulate steering angles to produce abrupt and directed intrusion cases. Note that six datasets have different ranges for steering angles. For example, for the HMB_6 sub-dataset of Udacity shown by Fig. 6(a), steering angles fall in [-0.17, 0.11] while the range for Chen_2018 is [-1.99, 0.55]. For each dataset, we apply appropriate values to modify the steering angles so as to make intrusion cases that are not trivial to be perceived.

Competitor We implement an IDS without considering the road context (referred to as IDS_wo_rc) for comparison. It is identical to the start-of-the-art IDS proposed by Taylor et al. [21]. IDS_wo_rc depends on learning CAN frames



(a) The Curve of Steering Angle over Time



(b) The Curve of Vehicle Speed over Time

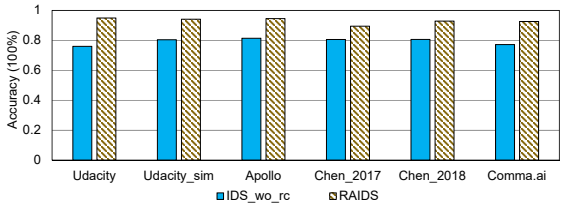
Fig. 6. Curves of Steering Angle and Vehicle Speed with Udacity's HMB_6 Dataset

with RNN to determine whether an arriving frame contains genuine data or not. For IDS_wo_rc, 70% of each dataset is used for training while 30% is for testing, the same as what we do with RAIDS. For both RAIDS and IDS_wo_rc, training is performed in a Linux server while testing is done in a Raspberry Pi 3 Model B+. Python 3.5 is installed in the server and Raspberry Pi. We assume that an embedded system with Raspberry Pi's computing powers is within the in-vehicle network gateway [11] where an IDS resides to protect the in-vehicle network.

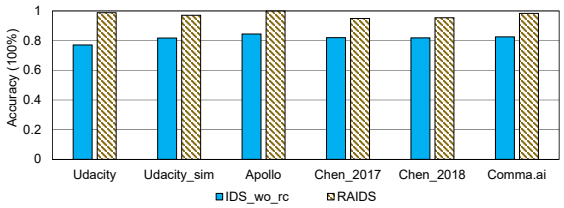
Metrics The main metric we use to compare RAIDS and IDS_wo_rc is the detection accuracy in testing, i.e., the ratio of detected normal and intrusion cases against overall cases. A higher detection accuracy means a better effectiveness of IDS. We also measure the ratios of undetected intrusion cases as well as false alarms by which an IDS wrongly labels a normal CAN frame to be anomalous. To study the efficiency of RAIDS, we record the average and maximum response time RAIDS spends in processing all cases of a dataset.

5.2 Detection Accuracy

Detection Accuracy Fig. 7 summarizes the detection accuracies of RAIDS and IDS_wo_rc with six datasets under abrupt and directed intrusions. The first observation obtained from Fig. 7 is that, RAIDS consistently achieves high detection accuracies across different datasets under both abrupt and directed intrusions. In particular, the highest accuracy for RAIDS is 99.9% with Apollo under directed intrusion while its lowest accuracy is 89.5% with Comma.ai under abrupt intrusion. IDS_wo_rc's highest accuracy is 84.5% with Apollo under directed intrusion while its lowest accuracy is 71.8% with Comma.ai under abrupt intrusion. The significant gap between RAIDS's and IDS_wo_rc's accuracies confirms the high effectiveness of RAIDS. RAIDS leverages the road context for intrusion detection while IDS_wo_rc solely relies on the data of historical CAN frames to apprehend the newly-arrived CAN frame. As shown in Fig. 6(b), the



(a) Abrupt Intrusion



(b) Directed Intrusion

Fig. 7. The Accuracies of Two IDSs under Two Types of Intrusions with Six Datasets

runtime volatile curve of steering angle alone is difficult to be modeled, unless it is associated with corresponding road context, like what RAIDS does. So the model built by IDS_wo_rc lacks reliability. RAIDS, nonetheless, extracts a feature vector of road context from each image and involves it for validating corresponding CAN frames. RAIDS thus establishes a sound model that maps a specific road context, like a road bend, to CAN frames. In summary, if adversaries put frames with abnormal data on the CAN bus, the unreliable model of IDS_wo_rc is ineffective in identifying the anomaly; however, on account of using road context, RAIDS has a much higher likelihood of detecting the intrusion.

The second observation obtained from Fig. 7 is that the detection accuracy under directed intrusion is consistently higher than that under abrupt intrusion, especially for RAIDS. For example, with Udacity, Apollo and Comma.ai datasets, the accuracy of RAIDS under directed intrusion is 4.0%, 5.4% and 7.2% higher than that under abrupt intrusion, respectively. As mentioned, directed intrusion should be more hazardous than abrupt intrusion because the former intends to incur a sudden change at a specific occasion onto the in-vehicle communications. Such a sudden change yet brings in more significant violation to ongoing road context, which exactly matches the capability of RAIDS and can be easily captured. This explains why RAIDS yields higher detection accuracy under directed intrusion. Also, as shown in Fig. 6(a), there exist dramatic increase and decrease of steering angle at runtime in reality. Consequently, IDS_wo_rc does not raise much difference in accuracy, i.e., at most 5.4% with Apollo.

Unreported Intrusions We also record the percentages of detected and unreported intrusions as well as detected normals and false alarms for six datasets under two types of intrusions. These results help us gain a deeper understanding of the accuracies of RAIDS and IDS_wo_rc. They are detailed in Fig. 8. Let us first focus on the percentages of detected intrusion cases as the percentage of false alarms are generally low. One observation is that, in all 16 diagrams, RAIDS

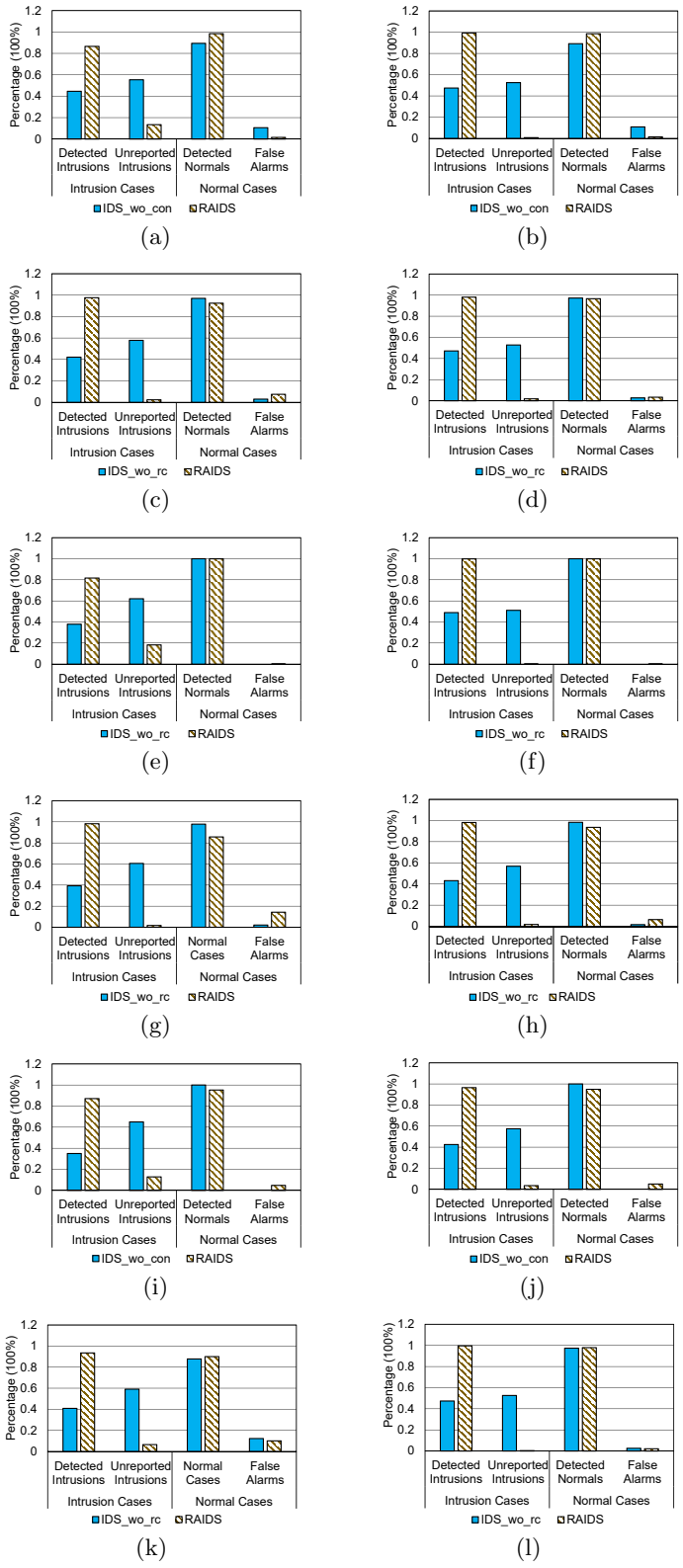
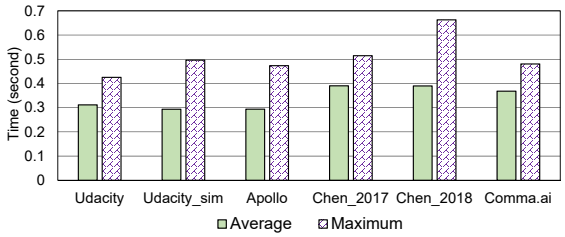
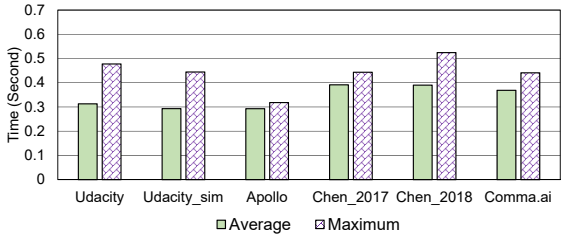


Fig. 8. The Percentages of Unreported Intrusions and False Alarms for Two IDSs under Abrupt and Directed Intrusions: (a) and (b) for Udacity; (c) and (d) for Udacity_sim; (e) and (f) for Apollo; (g) and (h) for Chen_2017; (i) and (j) for Comma.ai.



(a) Abrupt Intrusion



(b) Directed Intrusion

Fig. 9. The Average and Maximum Response Time of RAIDS under Two Intrusions

detects most of the intrusion cases while IDS_wo_rc even cannot report half of them. For example, in Fig. 8(i), the percentages of detected and unreported intrusion cases are 87.2% and 12.8%, respectively, for RAIDS with Chen_2018 under abrupt intrusion; however, they are 35.0% and 65.0% for IDS_wo_rc, respectively. In other words, without road context, IDS_wo_rc ignores many intrusion cases. This in turn justifies the importance of road context in intrusion detection. In addition, as to Apollo with the aforementioned 99.9% accuracy under directed intrusion, Fig. 8(f), tells that there is hardly unreported intrusion or false alarm. This reaffirms the highest accuracy achieved by RAIDS.

Second, let us make a comparison between abrupt and directed intrusions. Take Chen_2018 for example again with Fig. 8(i) and Fig. 8(j). The percentage of detected intrusion cases for RAIDS increases from 87.2% under abrupt intrusion to 96.5% under directed intrusion. Such an increase confirms that directed intrusion is likely to incur intrusion cases that are easier to be perceived. Comparatively, the percentage of detected intrusion cases for IDS_wo_rc jumps from 35.0% under abrupt intrusion to 42.5% under directed intrusion. Although the difference is considerable ($42.5\% - 35.0\% = 7.5\%$), it is less than that of RAIDS ($96.5\% - 87.2\% = 9.3\%$). These numbers agree with the second observation we have with Fig. 7, and explain why the accuracy of IDS_wo_rc does not increase as much as that of RAIDS from abrupt intrusion to directed intrusion.

5.3 Response Time

With Raspberry Pi, we have measured the response time of RAIDS in detecting intrusions for six datasets. The two diagrams in Fig. 9 capture the average

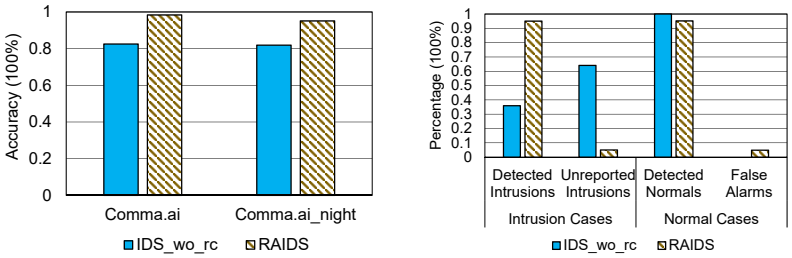


Fig. 10. A Comparison between Daytime and Nighttime Datasets for RAIDS: (a) Detection Accuracy and (b) Unreported Intrusions and False Alarms with Comma.ai_night

and maximum response time of processing all records of images and corresponding CAN frames under abrupt and directed intrusions, respectively. The results shown in Fig. 9 state that none of the average response time is greater than 0.40 second. More important, the maximum response time, which meas the worst-case response time, is mostly no greater than 0.52 second, except with Chen_2018 for which RAIDS cost 0.66 second under abrupt intrusion. Concretely, the short response time justifies the efficiency of RAIDS.

The average response time does not deviate much from the maximum response time because each record contains almost the same quantity of data, i.e., image and CAN frames, for RAIDS to handle. The marginal deviation is mainly caused by other running programs and system scheduling in a real embedded computing system. Note that we have used an economical Raspberry Pi for testing. The computational resources of Raspberry Pi include an inexpensive 1.4GHz ARM CPU and 1GB DRAM. With regard to employing RAIDS in real-world autonomous cars, a more powerful embedded system with high-end CPU and larger RAM space could be leveraged to reduce the response time, which surely improves the efficiency and applicability of RAIDS.

5.4 The Impact of Daytime and Night

It is non-trivial to extract meaningful features from nighttime images due to the generally low visibility of road conditions. Comma.ai provides a sub-dataset with nighttime images and CAN frames (referred to as Comma.ai_night). We have done abrupt and directed intrusions with it. Due to space limitation, we present the results under directed intrusion. Fig. 10(a) captures the comparison of IDS_wo_rc's and RAIDS's accuracies between daytime Comma.ai and nighttime Comma.ai_night. The accuracy of IDS_wo_rc does not fluctuate much since it is oblivious of the change of day and night. Nevertheless, due to the weaker perception of road context at night, the accuracy of RAIDS drops by 3.3%.

A comparison between Fig. 8(l) for Comma.ai and Fig. 10(b) for Comma.ai_night shows that, the percentages of unreported intrusions and false alarms for RAIDS increase by 4.6% and 2.7%, respectively. Thus, a bit more mistakes (undetected intrusions and false alarms) were made due to the relatively obscure nighttime road contexts. This explains the accuracy drop of RAIDS for nighttime images.

6 Conclusion

In this paper, we investigate how to effectively detect intrusions to the in-vehicle communications of autonomous car. In an autonomous car, a self-driving model reads sensory information reflecting ever-changing road contexts and generates control signals that are eventually transformed into CAN frames. We accordingly develop RAIDS. RAIDS extracts and abstracts road contexts from sensory information into a feature vector. It then leverages such an expressive feature vector of road context to assert the genuineness of observed CAN frames. We have built a prototype for RAIDS through lightweight neural networks, and evaluated it in an embedded computing system with extensive datasets. Experimental results confirm that RAIDS achieves up to 99.9% accuracy with short response time.

Acknowledgements. The authors appreciate the pioneering attempts done by Tanya Srivastava and Pryanshu Arora. This work is partially supported by the Ministry of Education of Singapore under the grant MOE2018-T2-1-098 and Singapore University of Technology and Design under the grant SRIS17123.

References

1. Apollo.auto: Roadhackers platform in Baidu Apollo project (April 2018), http://data.apollo.auto/static/pdf/road_hackers_en.pdf
2. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint: 1604.07316 (2016)
3. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX Conference on Security. pp. 77–92. USENIX Security '11, USENIX Association, Berkeley, CA, USA (2011)
4. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: DeepDriving: Learning affordance for direct perception in autonomous driving. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). pp. 2722–2730. ICCV '15, IEEE Computer Society, Washington, DC, USA (2015)
5. Chen, S.: Sully Chen's driving datasets (2017 & 2018) (April 2018), https://github.com/SullyChen/driving-datasets
6. Cho, K.T., Shin, K.G.: Fingerprinting electronic control units for vehicle intrusion detection. In: Proceedings of the 25th USENIX Security Symposium. pp. 911–927. USENIX Security '16, USENIX Association, Austin, TX (2016)
7. Chollet, F., et al.: Keras: Deep learning for humans (February 2019), https://keras.io
8. Chowdhury, A., Karmakar, G., Kamruzzaman, J., Saha, T.: Detecting intrusion in the traffic signals of an intelligent traffic system. In: Information and Communications Security. pp. 696–707. Springer International Publishing, Cham (2018)
9. Comma.ai: The Comma.ai driving dataset (October 2016), https://github.com/commaai/research
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (June 2016)

11. Hoppe, T., Kiltz, S., Dittmann, J.: Security threats to automotive CAN networks–practical examples and selected short-term countermeasures. Reliability Engineering & System Safety **96**(1), 11 – 25 (2011), special Issue on Safecomp 2008
12. Kang, M.J., Kang, J.W.: Intrusion detection system using deep neural network for in-vehicle network security. PloS one **11**(6), e0155781 (2016)
13. Kleberger, P., Olovsson, T., Jonsson, E.: Security aspects of the in-vehicle network in the connected car. In: 2011 IEEE Intelligent Vehicles Symposium (IV). pp. 528–533. IEEE (June 2011)
14. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al.: Experimental security analysis of a modern automobile. In: 2010 IEEE Symposium on Security and Privacy. pp. 447–462. IEEE (2010)
15. Müter, M., Asaj, N.: Entropy-based anomaly detection for in-vehicle networks. In: 2011 IEEE Intelligent Vehicles Symposium (IV). pp. 1110–1115 (June 2011)
16. Paszke, A., Gross, S., Chintala, S., Chanan, G.: PyTorch (February 2019), https://pytorch.org/
17. Petit, J., Shladover, S.E.: Potential cyberattacks on automated vehicles. IEEE Transactions on Intelligent Transportation Systems **16**(2), 546–556 (2015)
18. PyTorch: PyTorch BCELoss function (February 2019), https://pytorch.org/docs/stable/nn.html
19. Song, H.M., Kim, H.R., Kim, H.K.: Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: Proceedings of the 2016 International Conference on Information Networking (ICOIN). pp. 63–68. ICOIN '16, IEEE Computer Society, Washington, DC, USA (2016)
20. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation pp. 1–13 (2019)
21. Taylor, A., Leblanc, S., Japkowicz, N.: Anomaly detection in automobile control network data with long short-term memory networks. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA). pp. 130–139 (Oct 2016)
22. Udacity: Udacity's self-driving car simulator (July 2017), https://github.com/udacity/self-driving-car-sim
23. Udacity: The Udacity open source self-driving car project (April 2018), https://github.com/udacity/self-driving-car
24. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1–16 (May 2019)
25. Wasicek, A., Pesé, M.D., Weimerskirch, A., Burakova, Y., Singh, K.: Context-aware intrusion detection in automotive control systems (June 2017), 5th Embedded Security in Cars (ESCar '17)