Journal Pre-proof

A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems

Tejasvi Alladi, Varun Kohli, Vinay Chamola, F. Richard Yu, Fellow, IEEE

PII: S2352-8648(22)00140-7

DOI: https://doi.org/10.1016/j.dcan.2022.06.018

Reference: DCAN 472

To appear in: *Digital Communications and Networks*

Received Date: 6 May 2021

Revised Date: 21 June 2022

Accepted Date: 28 June 2022

Please cite this article as: T. Alladi, V. Kohli, V. Chamola, F.R. Yu, Fellow, IEEE, A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems, *Digital Communications and Networks* (2022), doi: https://doi.org/10.1016/j.dcan.2022.06.018.

Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems

Tejasvi Alladi[a], Varun Kohli[b], Vinay Chamola[*,b], F. Richard Yu[c], *Fellow, IEEE*

[a] **Department of Computer Science and Information Systems, BITS-Pilani, Pilani Campus, 333031, India**

[b] **Department of Electrical and Electronics Engineering & APPCAIR, BITS-Pilani, Pilani Campus, 333031, India**

[c] **Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada**

Abstract

With the rise of the Internet of Vehicles (IoV) and the number of connected vehicles increasing on the roads, Cooperative Intelligent Transportation Systems (C-ITSs) have become an important area of research. As the number of Vehicle to Vehicle (V2V) and Vehicle to Interface (V2I) communication links increases, the amount of data received and processed in the network also increases. In addition, networking interfaces need to be made more secure for which existing cryptography-based security schemes may not be sufficient. Thus, there is a need to augment them with intelligent network intrusion detection techniques. Some machine learning-based intrusion detection and anomaly detection techniques for vehicular networks have been proposed in recent times. However, given the expected large network size, there is a necessity for extensive data processing for use in such anomaly detection methods. Deep learning solutions are lucrative options as they remove the necessity for feature selection. Therefore, with the amount of vehicular network traffic increasing at an unprecedented rate in the C-ITS scenario, the need for deep learning-based techniques is all the more heightened. This work presents three deep learning-based misbehavior classification schemes for intrusion detection in IoV networks using Long Short Term Memory (LSTM) and Convolutional Neural Networks (CNNs). The proposed Deep Learning Classification Engines (DCLE) comprise of single or multi-step classification done by deep learning models that are deployed on the vehicular edge servers. Vehicular data received by the Road Side Units (RSUs) is pre-processed and forwarded to the edge server for classifications following the three classification schemes proposed in this paper. The proposed classifiers identify 18 different vehicular behavior types, the F1-scores ranging from 95.58% to 96.75%, much higher than the existing works. By running the classifiers on testbeds emulating edge servers, the prediction performance and prediction time comparison of the proposed scheme is compared with those of the existing studies.

KEYWORDS:
Vehicular Ad-hoc Networks (VANETs), Intelligent Transportation Systems (ITS), Artificial Intelligence (AI), Deep Learning, Internet of Things (IoT)

1. INTRODUCTION

Internet of Vehicles (IoV) is a novel vehicular network paradigm that has been extensively discussed in the literature. Due to the ever-increasing number of vehicles on the roads and the growing requirement for connected vehicles, IoV is projected as a crucial architecture for the futuristic Cooperative Intelligent Transportation Systems (C-ITSs), combining the benefits of the Internet of Things (IoT) and Vehicular Ad-hoc Networks (VANETs). It is expected to provide efficient and high bandwidth communication links owing to the enabling technologies deployed in

*Corresponding author
[1]Email addresses: tejasvi.alladi@pilani.bits-pilani.ac.in (T. Alladi), varun.kohli@u.nus.edu (V. Kohli), vinay.chamola@pilani.bits-pilani.ac.in (V. Chamola), richard.yu@carleton.ca (F. R. Yu)

IoV, such as Dedicated Short-Range Communication (DSRC) and 5G-based cellular Vehicle to Everything (V2X) communication [1]. However, the various advantages offered by IoV come along with cybersecurity challenges [2, 3]. The communication between vehicle On-board Units (OBUs) and Road Side Units (RSUs) is wireless in nature and susceptible to several types of known security attacks [4, 5]. With various types of entities such as OBUs and RSUs communicating in the network betweem each other, the number and type of vulnerable wireless interfaces also increases. DSRC and V2X are newly developed standards for communication between the various entities in the vehicular network that further increase the risk of unknown cyberattacks [6, 7, 8].

Several security techniques have been proposed in recent years for securing the OBU-RSU and OBU-OBU communication links in the vehicular networks, by leveraging advanced technologies such as blockchain [9, 10, 11], Artificial Intelligence (AI) [12, 13] and cryptographic schemes. Although these techniques are shown to be able to secure the network against various cybersecurity attacks, securing each possible link in the IoV networks, and ensuring privacy for each entity is a near-impossible task. Thus, there is a need to complement these security schemes with Intrusion Detection Schemes (IDSs). IDSs for intra-vehicular networks have been explored for many years. They rely on learning techniques such as information entropy-based detection [14], clock drift-based detection [15], bloom filtering [16], machine learning-based detection [17] and deep learning-based detection [18, 19]. However, to the best of our knowledge, the IDS for inter-vehicular networks is still a largely unexplored area. Some machine learning-based IDSs for inter-vehicular networks have also been proposed, however, not many deep learning-based schemes have been developed yet. With the amount of vehicular data increasing at an unprecedented rate and with various types of cyberattacks being launched [20], machine learning algorithms may not be practical due to the large amount of time and resources needed for feature engineering. Deep learning, on the contrary, does not require manual feature engineering, thus making it more suitable for IDSs in the next-generation IoV networks.

Vehicle OBUs are resource-constrained devices, thus running prediction tasks using deep learning on the OBUs may be computationally inefficient. Instead, the models may be deployed on the cloud servers, taking into account the limited resources of all types of vehicles. However, running tasks on cloud servers leads to high economic costs and latency. To address this issue, edge computing as an alternative to cloud services has been actively researched in the existing literature [21, 22, 23]. For instance, Hussain *et al.* [24] proposed an edge computing-based deep learning architecture for the detection of anomalies in 5G-based cellular networks. Using this solution, the data broadcast by a vehicle will be received by the RSUs, which in turn would forward the data to the edge servers where the deployed deep learning classifiers, which are a combination of one or more pre-trained deep learning models, would detect misbehavior types in the received data.

In this paper, we propose a deep learning-based misbehavior classification scheme for addressing the security issues in C-ITS applications such as IoV networks. We also present a comparison of the developed classifiers using popular comparison metrics. The major contributions of this paper are highlighted below:

i. We propose deep learning-based single-stage and two-stage classifiers of 18 vehicular classes (1 normal behavior class and 17 misbehavior classes) for intrusion detection in IoV networks. The classifiers, henceforth called Deep Learning Classification Engines (DLCEs) comprise coarse-grained and fine-grained classification stages.
ii. Various deep learning models are trained in this experiment to develop DLCEs. These DLCEs are capable of detecting and classifying different types of known vehicle misbehavior types in the IoV network.
iii. We use two popular testbeds, namely, Nvidia Jetson Nano and Raspberry Pi 3B, to emulate vehicular edge servers. The viability of deploying the DLCEs on the edge servers is discussed by conducting extensive experimentation for vehicle misbehavior prediction performance and prediction time results.

The rest of this paper is organized as follows. Section 2 discusses the related works on intrusion detection in vehicular networks. Section 3 gives a brief introduction to the concepts of deep learning used in this paper such as Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM). We present the proposed deep learning-based misbehavior classification and intrusion detection scheme in Section 4. The results for running various deep learning models and the performance analysis of the proposed classifiers are presented in Section 5. A follow-up discussion on the prediction performance and prediction time is presented in Section 6. The paper is finally concluded in Section 7.

2. Related work

Intrusion detection for vehicular networks is an active area of research, with various techniques being explored, such as statistical approach, cryptography, machine learning, and deep learning. Raya *et al.* proposed an entropy approach to model normal vehicle behavior, in which any deviation observed in vehicle movement is considered an anomaly [25]. However, this approach is inefficient in the IoV scenario

where the number of attacks is very large. Another approach based on learning automata was proposed in [26]. However, due to the high latency observed in this approach, it is not suitable for real-time applications such as vehicular networks. A statistical approach for intrusion detection in VANETs was proposed in [27], which is capable of detecting false information attacks in the network by identifying rogue nodes. A big data framework for network intrusion detection system was proposed in [28]. The classification algorithm in this scheme is based on Random Forests (RF), a type of machine learning technique for traffic detection. Zhang *et al.* proposed a collaborative and distributed intrusion detection scheme, in which the participating vehicles run machine learning models on the individual nodes and exchange information about anomalous behavior in the network [29]. Another machine learning-based intrusion detection model was presented in [1], which use Fuzzy C-means clustering and Elliptic Curve Cryptography (ECC) for preventing and detecting intrusions in vehicular networks. However, due to the large amount of vehicular data generated, traditional machine learning techniques cannot be completely relied on for intrusion detection in the IoV scenario.

In this regard, some deep learning-based security techniques have also been proposed recently. Van *et al.* [30] proposed a deep learning technique based on CNNs for the detection and identification of anomalous sensor data to prevent crashes in automated vehicles. However, their method is limited only to the anomaly types listed in the paper, and specific to vehicles with automation level 4 and level 5. A supervised learning approach for intrusion detection based on CNNs has also been proposed in the existing literature [31], however, the attacks considered in this approach are limited in number. Another work takes an unsupervised approach for anomaly detection of network traffic in VANETs [32]. Loukas *et al.* proposed a cloud-based IDS that demonstrates the benefit of running the deep learning-based intrusion detection tasks on the cloud [33]. Since vehicles are equipped with only limited processing power, offloading is necessary especially when DLCEs have to be run, however, offloading to the cloud servers can be inefficient in terms of cost, resource availability, and latency.

Edge servers have been discussed widely in the literature as an alternative to running tasks on the remote cloud servers [34, 35]. In recent studies, these edge servers have also been considered for task offloading in vehicular networks, where edge servers are co-located along with the RSUs. Ning *et al.* have implement a deep reinforcement algorithm for IoV networks which runs at the network edge and provides real-time responses to the vehicles [36]. This scheme is shown to be superior to the other schemes compared to their work. Chen *et al.* [37] proposed a vehicular edge computing architecture for deploying deep reinforcement techniques as part of an intelligent path planning scheme for platoons of autonomous vehicles. Some other studies combined the concepts of deep Q-learning and edge computing to achieve efficient caching in vehicular networks [38, 39]. Yu *et al.* proposed a federated learning-based edge caching scheme for meeting the demands of next-generation low latency and high computation applications for vehicular networks [40]. Proactive content caching at the vehicular network edge is a promising approach for achieving the requirements of high mobility, user privacy, and scalability in vehicular networks. Running deep learning-based services using CNNs and Genetic Algorithms on edge servers has also been studied in the literature for resource utilization in vehicular networks [41].

Based on the various studies discussed above, it can be seen that there is a need for developing a deep learning-based IDS that is well suited to the real-time and data-intensive applications of IoV. There is also a need for backing up these systems with resource-rich network infrastructure such as edge servers to meet the real-time latency requirements. This paper proposes such a deep learning-based classification approach based on edge computing.

3. Preliminary background

This section presents a brief background knowledge of deep learning techniques used in this paper. Deep learning imitates the ability of the human brain to process data and create patterns from it for making decisions and is capable of learning from data that may be unstructured and unlabeled. There are several deep learning models, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). Due to the time-series nature of vehicular data, CNN and LSTM prove to be the best choices among the available deep learning models. This study, therefore, uses the CNN and LSTM models for misbehavior classification for sequential vehicular data.

3.1. Convolutional neural network

A CNN is a hierarchical system that uses a sliding filter of fixed width on the input of each convolutional layer. The sliding filter shifts to collect new observations at the end of each epoch. The input may be of one dimension (1D) or higher. In the present study, we work with 1D convolutional layers. A CNN comprises an input layer, multiple hidden layers, and an output layer. The hidden layers are a series of convolutional layers, followed by normalization, pooling, or fully connected layers. Each neuron in the convolutional layer operates on input X to obtain the output a by the following equation:

$$a^{(l)} = f(X^{(l-1)} \otimes w^{(l-1)} + b^{(l-1)}) \quad (1)$$

Here, w is the weight vector, and b is a scalar corresponding to the bias of the neuron. The function $f(z)$ is the activation function which can be among Rectified Linear Unit (ReLU), sigmoid, hyperbolic tangent and so on. During the training process (backpropagation), w and b are set to values such that the DL model can fit the relationship between outputs and inputs. For this, a loss function L is defined to calculate the closeness of the predicted output to the real output, and the backpropagation algorithm minimises the loss for a given training example $X^{(m)}, y^{(m)}$ by using the gradient descent algorithm. The partial derivatives in the gradient descent algorithm denoted as $\frac{\partial L}{\partial w_{i,j}^{(l)}}$ and $\frac{\partial L}{\partial b_i^{(l)}}$ are computed, where $w_{i,j}^{(l)}$ corresponds to the weight for the connection between the i^{th} neuron of the $(l+1)^{th}$ layer and the j^{th} neuron of the l^{th} layer, and $b_i^{(l)}$ is the bias associated with the i^{th} neuron in the $(l+1)^{th}$ layer. The loss function for M training examples is defined as:

$$L(w, b) = 1/m \sum_{m=1}^{M} (L(w, b, X^{(m)}, y^{(m)})) \quad (2)$$

The weights and biases are updated with a learning rate α according to the following equations:

$$w_{i,j}^{(l)} = w_{i,j}^{(l)} - \alpha \frac{\partial L}{\partial w_{i,j}^{(l)}} \quad (3)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial L}{\partial b_i^{(l)}} \quad (4)$$

3.2. Long short term memory

LSTMs are a type of RNN capable of keeping sequential information intact. RNNs outperform ANNs for sequential data but also suffer from vanishing gradient problems. This hampers the ability of the network to remember information over long periods since any two important events in the time series can occur with large gaps. LSTMs are developed to solve the problem of vanishing gradients with their relative insensitivity to this gap length. LSTM networks are best suited for classification problems for time series data due to the ability of the cells to remember values over arbitrary time intervals.

LSTMs have a gated structure and use ANN units. They can store information across time steps without drastic changes, thus forming a long-term memory. The short-term working memory is updated more often. LSTM units consist of a cell and the flow of information in and out of the cell is regulated by sigmoid activated neural networks called gates. The three gates in an LSTM unit are as follows:

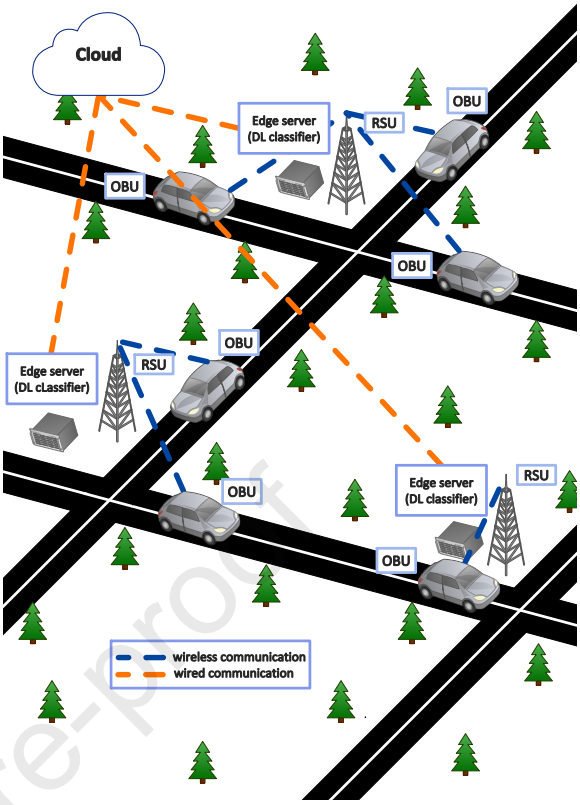

Fig. 1: IoV Network model.

1. **Save gate:** It regulates the information from the current input that is added to the cell state.

$$s_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \quad (5)$$

2. **Forget gate:** It controls what information the cell forgets.

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \quad (6)$$

3. **Output gate:** It controls what information is given as output.

$$o_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \quad (7)$$

The memory states are updated as:

$$\tilde{c}_t = f(W_c[h_{t-1}, X_t] + b_c) \quad (8)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \quad (9)$$

$$h_t = o_t \cdot f(c_t) \quad (10)$$

Wherein + and $\cdot$ are element-wise addition and element-wise multiplication respectively, and f denotes the activation function.

4. Proposed misbehavior classification scheme for intrusion detection

This section initially presents the IoV network model, the dataset considered, and the data preprocessing steps involved. Further, the classifiers developed for the proposed misbehavior classification scheme and their training/testing methodology are discussed.

4.1. Network model

We consider a network of vehicles, with vehicles exchanging data with each other and with the RSUs, deployed at major intersections on the roads. The vehicle OBUs can communicate with the RSUs (V2I communication) using either DSRC or cellular V2X communication technologies. Each RSU in the network is co-located with an edge server on which the DLCEs are deployed. The edge servers are connected to the cloud servers through high-speed wired backhaul communication links. The deep learning models corresponding to the DLCEs are trained on the cloud servers, while the actual vehicle misbehavior prediction tasks are run on the edge servers. Whenever the data is broadcasted by a vehicle, it is received by the nearest RSU. The RSU passes it onto the edge server to check for possible misbehavior. This network model is depicted in Fig. 1.

The vehicles can either exhibit normal behavior or any type of misbehavior. Misbehavior is categorized into a fault type or a possible attack type. The data used in this work comes from a popular dataset for misbehavior detection in vehicular networks called *VeReMi Extension* [42]. This dataset covers a wide range of misbehavior types, with faults pertaining to incorrect position and velocity values, and cyberattacks such as Denial of Service (DoS) attack, disruptive attack, replay attack, and their variants [43].

4.2. Data preprocessing

4.2.1. Structure of the dataset

The *VeReMi Extension* dataset consists of 24 simulations, one for each hour of the day from hour 0 to hour 23. Each simulation consists of the original trace logs of each vehicle that traversed the network in that particular hour, alongside one Ground Truth (GT) file containing all the data received by the edge servers during that hour. The data in these log files as well as the GT file of each simulation comprises fields ranging from the send time, sender ID, sender pseudo ID, message ID, and position-based information which includes position, velocity, acceleration, and heading along with the noise in each of these values. Since our study is on misbehavior detection at the level of edge servers, we used the 24 GT files for creating our dataset.

4.2.2. Field selection and labeling

Among the various data fields present in the GT file, the following are required for our study: sender ID, sender pseudo ID, send time, and two X, Y coordinate values of each position, velocity, acceleration, and heading. The GT files do not contain any misbehavior labels for the vehicle entries present in them. This information is available from the vehicle-wise log files for the corresponding simulation hour. To create a reduced vehicle-wise dictionary containing the required data of each vehicle along with the labels, we scanned through the GT files of all the simulation hours and selected the required fields and the given misbehavior labels from the log file names corresponding to each given vehicle.

4.2.3. Sequence creation

From the reduced vehicle data dictionary, multiple time sequences, each having 20 data points, were further created. Each data point consists of 7 values: label, send time, pseudo ID, X and Y position coordinates, and X and Y velocity coordinates. The values for acceleration and heading were left out since all the faults and attacks simulated in the base *VeReMi Extension* dataset are for the position, velocity, time, and pseudo IDs. These 20x7 sequences were generated with a slide length of 10, which implies that the next sequence is 10 data points after the start of the previous sequence, thus creating an overlap of 10 datapoints between one sequence and the next. Sequences with lengths lower than 20 were ignored. This means that if a vehicle consists of 55 data points, we get 4 sequences 0-20, 10-30, 20-40, 30-50, and the remaining incomplete sequence 40-55 is left out. The input dataset aims to have low repetition while containing a large sample size. Choosing a slide length has a trade-off between the two aforementioned aims, i.e. choosing a smaller slide length will create more repetition and more data, whereas, a larger slide length produces lower repetition and less data. Thus a slide length of 10 was chosen to lower the repetition while having a higher number of training samples. Once all the time sequences of a vehicle are generated, they are added to their corresponding misbehavior class in the misbehavior class-wise dictionary.

4.2.4. Misbehavior class selection

In this classification work, we attempt to classify 18 different class types mentioned in the original *VeReMi Extension* dataset. As can be seen in Table 1, this includes one normal vehicle behavior type, 8 fault types, and 9 attack types. The table also lists the number of sequences created for each class type.

Among the faults, types 1 to 4 correspond to positional misbehavior wherein a vehicle broadcasts its position as constant or random, as seen in the constant position (type 1) and random position (type 3) faults. In the constant and random position offsets (type 1 and 4 respectively), the vehicle broadcasts its positional data by adding or subtracting constant and random offsets respectively. Along similar lines, faults 5 to 8 correspond to velocity related misbehavior wherein type 5 and type 7 consist of constant and randomized vehicular velocity data, while type 6 and type 8 consist of vehicular data offset by constant and random offsets respectively added to or subtracted from the real velocity of the vehicle.

The attack types 9 to 17 are a combination of the four major attacks: DoS, Sybil, Data Replay, and Dis-

Table 1: Details of different vehicle types considered in this misbehavior classification schemes

Type	Class	Description	Total Seq.
0	Normal	Normal behavior	165373
1	Fault	Constant position	3804
2	Fault	Constant position offset	3793
3	Fault	Random position	3821
4	Fault	Random position offset	3701
5	Fault	Constant speed	3623
6	Fault	Constant speed offset	3886
7	Fault	Random speed	3662
8	Fault	Random speed offset	3705
9	Attack	Disruptive	3776
10	Attack	Data replay	3870
11	Attack	DoS	12564
12	Attack	DoS random	12103
13	Attack	DoS disruptive	12370
14	Attack	Data replay sybil	16981
15	Attack	Traffic congestion sybil	3876
16	Attack	DoS random sybil	8122
17	Attack	DoS disruptive sybil	7654

ruptive. DoS attacks are characterized by the transmission of data at a frequency higher than the network limit, thereby denying the operational services offered by the infrastructure to the vehicular nodes. They can therefore be identified by examining message frequencies. In Sybil attacks, a vehicle assumes multiple pseudo IDs, thereby gaining an advantage in the network. As the IDs are valid, it becomes difficult to identify such misbehavior. Data replay attacks occur when the attacking vehicle replays the data obtained from another vehicle. It is challenging to identify such attacks since the vehicular data replayed has patterns of a normal vehicle without changes in transmission frequency or the usage of multiple pseudo IDs. Disruptive attacks are a type of data replay wherein the attacker replays the messages received from multiple vehicles at random. This causes the network to be flooded by redundant data, preventing the broadcast of genuine information. The other attacks shown in Table 1 are a combination of these four major attack types.

4.3. Classifiers

In this section, we present three different types of classifiers (DLCEs) for the classification and detection of possible intrusion in the IoV network.

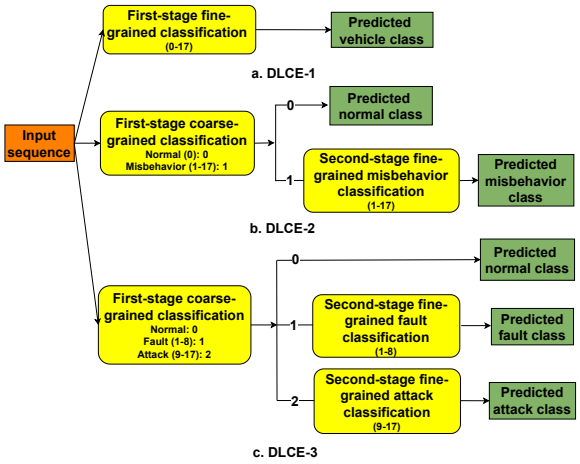

Fig. 2: Proposed classifiers (DLCEs) in this work.

4.3.1. DLCE-1

DLCE-1 is a single-stage fine-grained classifier wherein the deep learning model specifically predicts and classifies the input sequence into a normal type, or one of the 8 fault types, or one of the 9 attack types considered.

4.3.2. DLCE-2

DLCE-2 is a two-stage classifier where the first stage is a coarse-grained classification to distinguish the input sequence as normal or misbehavior. Based on this prediction, in the next stage the predicted misbehavior sequence is sent to the fine-grained deep learning model which predicts the specific fault or attack type.

4.3.3. DLCE-3

Being a two stage classifier, DLCE-3 works along similar lines as DLCE-2. The coarse-grained stage predicts the input sequence as a normal, fault, or attack type. Based on this prediction, the sequence is then sent to either a fine-grained fault or fine-grained attack prediction model for a more specific classification.

These three classifiers are depicted pictorially in Fig. 2.

4.4. Training and testing

For training the deep learning models, Mean Absolute Error ($\mathcal{MAE}$) was chosen as the loss function, as shown in Eq. 18, where $\hat{y}$ denotes the predicted values and n is the number of data points. We used the Adam optimizer with a learning rate of 0.0003. The convolutional and dense layers are activated with the ReLU activation function. For models comprising of one or more LSTM layers, the final LSTM layer has its return sequences parameter set to False whereas the ones before it has the parameter set to True.

$$\mathcal{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}| \tag{11}$$

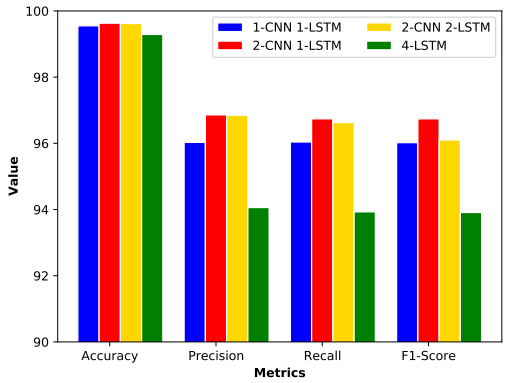

Fig. 3: Average performance metrics for single-stage fine-grained classification of DLCE-1.

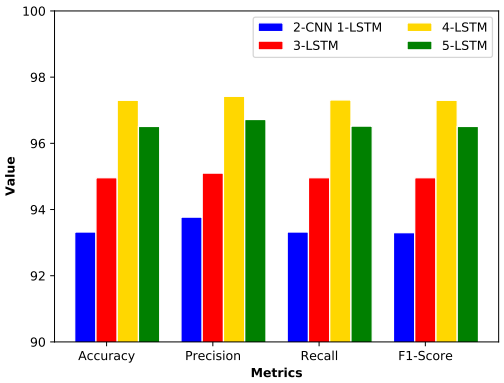

Fig. 4: Average performance metrics for the first-stage coarse-grained classification of DLCE-2.

Accuracy ($\mathcal{A}$), precision ($\mathcal{P}$), recall ($\mathcal{R}$) and F1-score ($\mathcal{F}1$) are defined in Eq.12, Eq.13, Eq.14 and Eq. 15 respectively. Here, $\mathcal{TP}$ means True Positive; $\mathcal{TN}$ means True Negative; $\mathcal{FP}$ means False Positive; and $\mathcal{FN}$ means False Negative.

$$\mathcal{A} = \frac{\mathcal{TP} + \mathcal{TN}}{\mathcal{TP} + \mathcal{FP} + \mathcal{TN} + \mathcal{FN}} \tag{12}$$

$$\mathcal{P} = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FP}} \tag{13}$$

$$\mathcal{R} = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FN}} \tag{14}$$

$$\frac{2\mathcal{PR}}{\mathcal{P} + \mathcal{R}} \tag{15}$$

5. Results

In this section, we present the prediction performance of the three DLCEs on the dataset considered. Based on extensive experimentation with various models, six deep learning models were finalized for performance evaluation for the three DLCEs considered. These models are *3-LSTM, 4-LSTM, 5-LSTM, 1-CNN 1-LSTM, 2-CNN 1-LSTM, and 2-CNN 2-LSTM* where *3-LSTM, 4-LSTM, 5-LSTM* are stacked LSTM models each with 3, 4, and 5 layers of 256 units respectively; *1-CNN 1-LSTM* has 1 layer of CNN and LSTM each with 512 units; *2-CNN 1-LSTM* has 2 layers of CNN with 1024, 512 units and 1 layer of LSTM with 512 units; and *2-CNN 2-LSTM* has 2 layers of CNN with 1024, 512 units and 2 layers of LSTM each with 256 units.

5.1. DLCE-1

5.1.1. Single-stage fine-grained classification

DLCE-1 takes a single-stage fine-grained classification approach on all the classes considered, including normal vehicle behavior, all faults, and all attacks.

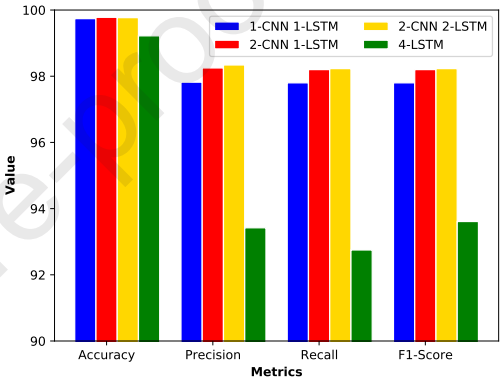

Fig. 5: Average performance metrics for the second-stage fine-grained classification of DLCE-2.

The performance metrics derived after running different deep learning models for DLCE-1 are presented in Table 2. On considering the macro-average (referred to as average in this study) of the metrics for each model as shown in Fig. 3, it can be seen that the *2-CNN 1-LSTM* model performs the best for DLCE-1 with the average values of accuracy, precision, recall, and F1-score being 99.63%, 96.86%, 96.74%, and 96.74% respectively. While this model performs well in most cases, its F1-score is low for types 0 (normal), 2 (constant position offset), 9 (disruptive), and 10 (data replay) among which types 0 and 2, as well as types 9 and 10 were cross-classified due to similarity in the behavior of these misbehavior types.

5.2. DLCE-2

5.2.1. First-stage coarse-grained classification

In DLCE-2, the first stage is a coarse-grained classification with type 0 being classified as class 0, and types 1-17 (all faults and attack types included) being classified as class 1. Table 3 summarizes the results of this classification for all the deep learning models considered. The average performance metrics for the first-stage coarse-grained classification of DLCE-2 are presented in Fig. 4. It can be observed that the model

Table 2: Performance metrics for single-stage fine-grained classification of DLCE-1

	1-CNN 1-LSTM				2-CNN 1-LSTM				2-CNN 2-LSTM				4-LSTM			
Type	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
0	99.01	89.24	94	91.56	98.75	85.45	94	89.52	98.23	78.77	94	85.71	97.8	76.57	89.33	82.46
1	99.16	93.84	91.33	92.57	99.5	97.9	93.33	95.56	99.47	100	90.67	95.1	98.72	93.98	83.33	88.34
2	98.78	92.75	85.33	88.89	99.05	94.96	88	91.35	98.6	91.24	83.33	87.11	98.37	89.71	81.33	85.31
3	99.85	97.4	100	98.68	99.81	96.77	100	98.36	99.96	99.34	100	99.67	99.57	93.71	99.33	96.44
4	99.62	96.67	96.67	96.67	99.81	100	96.67	98.31	99.77	100	96	97.96	99.26	94.56	92.67	93.6
5	99.62	97.95	95.33	96.62	99.54	98.59	93.33	95.89	99.62	100	93.33	96.55	98.95	94.89	86.67	90.59
6	99.73	96.73	98.67	97.79	99.77	98	98	98	99.69	95.51	99.33	97.39	99.3	92.31	96	94.12
7	99.77	96.15	100	98.04	99.54	92.59	100	96.15	99.73	95.54	100	97.72	99.72	95.54	100	97.72
8	99.92	98.68	100	99.34	100	100	100	100	99.96	99.38	100	99.67	99.65	95.48	98.67	97.05
9	98.56	88.36	86	87.16	98.94	86.34	96	91.14	99.31	92.31	96	84.12	98.29	87.32	82.67	84.93
10	98.59	87.41	88	87.71	98.98	96.21	85.33	90.46	99.35	96.5	92	94.2	98.03	82.35	84	83.17
11	100	100	100	100	100	100	100	100	99.92	99.33	99.33	99.33	100	100	100	100
12	99.92	100	98.67	99.33	99.96	100	99.33	99.67	99.89	99.33	98.67	99	99.96	100	99.33	99.67
13	99.88	98.04	100	99.01	99.96	99.34	100	99.67	99.92	98.68	100	99.34	99.88	98.04	100	99.01
14	100	100	100	100	100	100	100	100	99.96	100	99.33	99.67	99.92	100	98.67	99.33
15	99.85	98.67	98.67	98.67	100	100	100	100	100	100	100	100	100	100	100	100
16	99.85	98.03	99.33	98.68	99.85	98.01	99.33	98.67	99.85	97.4	100	98.68	99.92	98.68	100	99.34
17	99.73	98.64	96.67	97.64	99.85	99.32	98	98.66	99.85	100	97.33	98.65	99.92	100	98.67	99.33
Average	99.55	96.031	96.037	96.02	**99.63**	**96.86**	**96.74**	**96.74**	99.62	96.85	96.63	96.1	99.29	94.06	93.93	93.91

Table 3: Performance metrics for the first-stage coarse-grained classification of DLCE-2

	2-CNN 1-LSTM				3-LSTM				4-LSTM				5-LSTM			
Class	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
0	93.32	89.3	98.42	93.64	94.96	92.6	97.74	95.1	97.3	95.04	99.82	97.37	96.51	93.59	99.87	96.63
1	93.32	98.24	88.21	92.96	94.96	97.6	92.18	94.82	97.3	99.81	94.79	97.23	96.51	99.86	93.16	96.39
Average	93.32	93.77	93.32	93.3	94.96	95.1	94.96	94.96	**97.3**	**97.425**	**97.31**	**97.3**	96.51	96.725	96.52	96.51

Table 4: Performance metrics for the second-stage fine-grained classification of DLCE-2

	1-CNN 1-LSTM				2-CNN 1-LSTM				2-CNN 2-LSTM				4-LSTM			
Type	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
1	99.52	97.26	94.67	95.95	99.64	99.3	94.67	96.93	99.64	99.3	94.67	96.93	98.51	88.51	87.33	87.92
2	99.44	96.58	94	95.27	99.72	97.35	98	97.67	99.88	98.04	100	99.01	98.72	87.9	92	89.9
3	99.88	98.68	99.33	99	99.92	98.68	100	99.34	99.92	98.68	100	99.34	99.58	95.45	98	96.71
4	99.48	94.77	96.67	95.71	99.92	100	98.67	99.33	99.96	100	99.33	99.67	98.31	89.21	82.67	85.81
5	99.52	97.62	94	95.92	99.64	99.3	94.67	96.93	99.29	100	88	93.62	99.25	96.48	91.33	93.84
6	99.56	93.71	99.33	96.44	99.48	92.55	99.33	95.82	99.92	98.68	100	99.34	99.05	90.45	94.67	92.51
7	99.88	98.04	100	99.01	99.88	98.04	100	99.01	99.21	88.24	100	93.75	99.75	96.15	100	98.04
8	100	100	100	100	99.96	99.34	100	99.67	100	100	100	100	99.42	94.16	96.67	95.39
9	99.4	97.2	92.67	94.88	99.25	97.12	90	93.43	99.56	95.42	97.33	96.37	97.55	80.3	70.67	79.45
10	99.4	93.55	96.67	95.08	99.25	91.19	96.67	93.85	99.6	97.94	95.33	96.62	97.19	74.64	68.67	77.07
11	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
12	99.96	100	99.33	99.67	99.96	100	99.33	99.67	99.6	100	99.33	99.67	100	100	100	100
13	99.92	98.68	100	99.34	99.96	99.34	100	99.67	99.92	99.33	99.33	99.33	99.92	98.68	100	99.34
14	99.96	100	99.33	99.67	100	100	100	100	99.96	99.34	100	99.67	99.92	100	98.67	99.33
15	100	100	100	100	99.88	98.04	100	99.01	100	100	100	100	100	100	100	100
16	99.8	96.77	100	98.36	100	100	100	100	99.8	96.77	100	98.36	99.75	96.15	100	98.04
17	99.8	100	96.67	98.31	99.88	100	98	98.99	99.8	100	96.67	98.31	99.75	100	96	97.96
Average	99.74	97.82	97.8	97.8	**99.78**	**98.34**	**98.23**	**98.23**	99.77	98.25	98.2	98.2	99.22	93.42	92.75	93.61

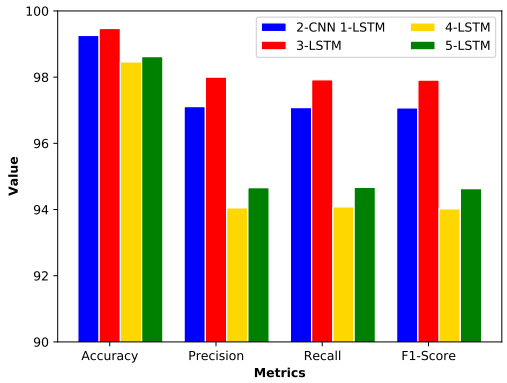

Fig. 6: Average performance metrics for the first-stage coarse-grained classification of DLCE-3.

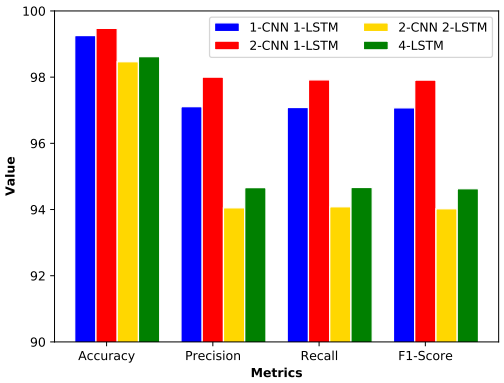

Fig. 7: Average performance metrics for the second-stage fine-grained fault classification of DLCE-3.

4-LSTM model performs the best with the accuracy, precision, recall, and F1-score being 97.3%, 97.42%, 97.31%, and 97.3% respectively. This model provides consistent predictions for both classes 0 and 1.

5.2.2. Second-stage fine-grained classification

The second stage is a fine-grained classification of each attack and fault type. Unlike DLCE-1, this classification does not consider the normal behavior class since it has already been filtered out in the first stage. From Table 4 we can observe that the average of each metric is higher compared to those found in DLCE-1. This can be attributed to the fact that since type 2 (constant position offset) closely resembles type 0 (normal behavior), type 2 is classified better by excluding type 0. The average metric comparison of the various models for this stage is depicted in Fig. 5. It can be observed that the average accuracy is the highest for the *2-CNN 1-LSTM* model with a value of 99.78%, whereas the average precision, recall, and F1-score are the highest for the *2-CNN 2-LSTM* model with values of 98.34%, 98.23%, and 98.23% respectively, making it the overall better choice. Also, this model performs better than the *2-CNN 1-LSTM* model of DLCE-1 of fault and attack classification. Although classes 9 (disruptive) and 10 (data replay) are still cross-classified, their performance is better. The lowest F1-scores among all the classes can be observed for classes 5 (constant speed) and 7 (random speed), which are cross-classified.

5.3. DLCE-3

5.3.1. First-stage coarse-grained classification

The first stage of DLCE-3 is a coarse-grained classification with type 0 being classified as class 0, types 1-8 (all faults) being classified as class 1, and types 9-17 (all attacks) being classified as class 2. Table 5 summarizes the results of the coarse-grained classification after running several deep learning models. The average metric comparison for this stage of DLCE-3 is

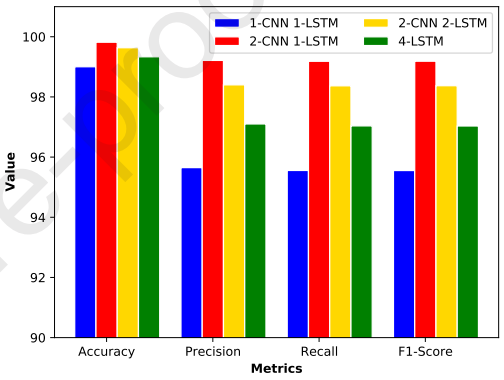

Fig. 8: Average performance metrics for the second-stage fine-grained attack classification of DLCE-3.

shown in Fig. 6. Similar to our observation in coarse-grained classification for DLCE-2, the *4-LSTM* model performs the best with the average accuracy, precision, recall, and F1-score of 98.58%, 97.9%, 97.87%, and 97.87% respectively. In comparison with the coarse-grained classification of DLCE-2, it is evident that the coarse-grained classification of DLCE-3 performs marginally better. Besides, the F1-score obtained for the classification of attacks is significantly higher than that obtained for normal behavior and faults which is due to the cross-classification of the normal behavior and the constant position offset present as a sub-class in the faults.

5.3.2. Second-stage fine-grained fault classification

One of the parallel second-stage fine-grained classifications is to classify the input into each fault type. Table 6 displays the individual metrics for each class and the average of each metric for the fine-grained fault classification. Fig. 7 shows the comparison of average metrics of the different models. The *2-CNN 1-LSTM* model can be seen to perform the best for this classification with the average accuracy, precision, recall, and F1-score being 99.47%, 98%, 97.92%, and 97.91% respectively. The lowest F1-scores obtained

Table 5: Performance metrics for the first-stage coarse-grained classification of DLCE-3

	2-CNN 1-LSTM				3-LSTM				4-LSTM				5-LSTM			
Class	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
0	92.34	82.85	97.25	89.48	93.94	86.12	97.69	91.54	98.18	95.82	98.87	97.32	93.41	85.24	97.12	90.8
1	92.6	97.56	79.94	87.87	93.88	97.46	83.94	90.19	98.14	98.71	95.69	97.17	93.53	96.88	83.37	89.62
2	99.17	98.5	99.18	98.84	99.05	98.44	98.87	98.65	99.41	99.18	99.06	99.12	99.38	99.12	99.12	99.12
Average	94.7	92.97	92.12	92.06	95.62	94.01	93.5	93.46	**98.58**	**97.9**	**97.87**	**97.87**	95.44	93.75	93.2	93.18

Table 6: Performance metrics for the second-stage fine-grained fault classification of DLCE-3

	1-CNN 1-LSTM				2-CNN 1-LSTM				2-CNN 2-LSTM				4-LSTM			
Type	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
1	98.73	97.87	92	94.85	99.16	100	93.33	96.55	98.52	97.84	90.67	94.12	97.34	93.43	85.33	89.2
2	98.81	94.16	96.67	95.39	99.66	99.32	98	98.66	96.41	88.03	83.33	85.62	97.59	89.1	92.67	90.85
3	99.66	97.4	100	98.68	99.66	97.4	100	98.68	99.21	94.9	99.33	97.07	99.65	98.03	99.33	98.67
4	98.98	96.62	95.33	95.97	99.83	99.33	99.33	99.33	97.16	89.8	88	88.89	97.68	92.41	89.33	90.85
5	98.98	96.62	95.33	95.97	98.82	97.89	92.67	95.2	97.83	91.39	92	91.69	98.18	92.16	94	93.07
6	99.49	98.65	97.33	97.99	99.75	98.04	100	99.01	99.12	94.3	99.33	96.75	99.13	96.67	96.67	96.67
7	99.4	95.54	100	97.72	98.91	92.02	100	95.85	99.47	96.15	100	98.04	99.47	96.15	100	98.04
8	100	100	100	100	100	100	100	100	100	100	100	100	99.91	99.34	100	99.67
Average	99.26	97.11	97.08	97.07	**99.47**	**98**	**97.92**	**97.91**	98.46	94.05	94.08	94.02	98.62	94.66	94.67	94.63

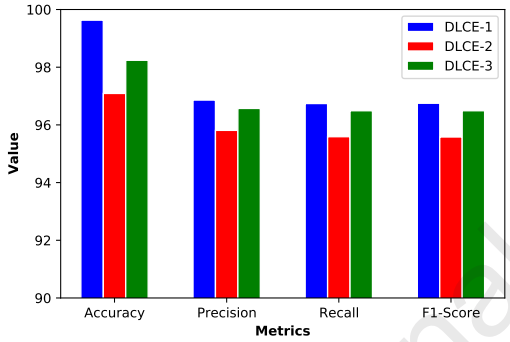

Fig. 9: Overall performance comparison of the three classifiers (DLCEs).

are for types 5 (constant speed) and 7 (random speed), which are cross-classified.

5.3.3. *Second-stage fine-grained attack classification*

The other parallel second-stage fine-grained classification stage is to classify the input into each attack type. Table 7 displays the individual metrics for each class and the average of each metric for the fine-grained attack classification. The average metric comparison for the models tested in this stage is shown in Fig. 8. It can be seen that the *2-CNN 1-LSTM* model performs the best with the average accuracy, precision, recall, and F1-score being 99.82%, 99.21%, 99.18%, and 99.19% respectively. We can also observe that the metrics for the *2-CNN 1-LSTM* model are better for fine-grained attack classification compared to fine-grained fault classification. Also, there is some cross-classification between classes 9 (disruptive) and 10 (data replay).

6. Discussion

We presented three different types of DLCEs to classify the input temporal sequences into normal behavior, faults or attacks. The overall average metric M_{avg} for DLCE-1 is calculated from the results as follows:

$$M_{avg} = \sum_{n=0}^{17} m_n \tag{16}$$

where m_n is any of the metrics $\mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{F}1$ for the n^{th} class (out of total 17 classes) of DLCE-1. The overall average metric M_{avg} for DLCE-2 is calculated from the results as follows:

$$M_{avg} = \sum_{n=0}^{1} m_n^{(1)} * \sum_{l=1}^{17} m_l^{(2)}/100 \tag{17}$$

where $m_n^{(1)}$ and $m_l^{(2)}$ are the n^{th} and l^{th} class metrics for coarse-grained classification (superscript (1)) and fine-grained classification (superscript (2)) respectively. The method for DLCE-3 follows a similar approach as DLCE-2:

$$M_{avg} = \sum_{n=0}^{1} m_n^{(1)} * (8 * \sum_{l=1}^{8} m_l^{(2)} + 9 * \sum_{k=9}^{17} m_k^{(2)})/1700 \tag{18}$$

Table 8 shows the overall performance of the three DLCEs based on the average performance metrics calculated for each stage of the classifiers. The table also provides a performance comparison of the proposed scheme with the existing studies on the same dataset [43, 44]. It can be observed that all the three proposed DLCEs outperform the methods used in the existing studies across all four metrics. The average performance metrics for the DLCEs calculated from the above equations are also plotted in the bar graph for comparison in Fig. 9. DLCE-1 shows the best results, while DLCE-2 shows the smallest average values among the three classifiers. The results for DLCE-3 closely follow those of DLCE-1. The shortcoming

Table 7: Performance metrics for the second-stage fine-grained attack classification of DLCE-3

	1-CNN 1-LSTM				2-CNN 1-LSTM				2-CNN 2-LSTM				4-LSTM			
Type	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
9	96.13	86.03	78	81.82	99.55	96.15	100	98.04	98.81	92.4	97.33	94.81	97.76	87.5	93.33	90.32
10	96.05	79.39	87.33	83.17	99.55	100	96	97.96	98.74	96.5	92	94.2	97.76	92.86	86.67	89.65
11	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
12	99.85	100	98.67	99.33	100	100	100	100	100	100	100	100	100	100	100	100
13	99.85	98.68	100	99.34	100	100	100	100	100	100	100	100	99.7	97.4	100	98.68
14	100	100	100	100	100	100	100	100	100	100	100	100	99.7	100	97.33	98.65
15	99.61	97.39	99.33	98.35	99.93	100	99.33	99.67	99.92	100	99.33	99.67	100	100	100	100
16	99.92	99.36	100	99.67	99.63	96.77	100	98.36	99.62	97.39	99.33	98.35	99.54	96.15	100	98.04
17	99.61	100	96.67	98.3	99.7	100	97.33	98.65	99.62	99.32	97.33	98.32	99.54	100	96	97.96
Average	99	95.65	95.56	95.55	**99.82**	**99.21**	**99.18**	**99.19**	99.63	98.4	98.37	98.37	99.33	97.1	97.04	97.03

Table 8: Performance comparison of the three classifiers (DLCEs) with existing works

	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}1$
DLCE-1	99.63	96.86	96.74	96.75
DLCE-2	97.09	95.81	95.59	95.58
DLCE-3	98.24	96.57	96.49	96.49
[43]	92.93	99.12	82.28	89.92
[44]	-	94.5	91.77	93

Table 9: Run-time results (in milliseconds) for the deep learning models used

	4-LSTM		2-CNN 1-LSTM	
Time	**Nano**	**Rpi**	**Nano**	**Rpi**
Data setup	287.74	570.12	251.24	336.84
Prediction	0.33	0.357	0.32	0.319
Total	288.07	570.48	251.56	337.16

of 3.25% to 4.42% F1 among the overall classification of the three proposed DLCEs arises from the similarities among faults such as constant position offsets (type 2) and random position offsets (type 4) with normal behavior (type 0) and random position behavior (type 3) respectively, and attacks such as disruptive (type 9) and data replay (type 10). This leads to inter-classification by each of the three DLCEs, thereby degrading the performance as seen in Section 5. This is a limitation of the learning models while classifying data that has similar patterns. But it can be argued that better performance is obtained from individual classification of faults and attacks in DLCE-3, unlike in DLCE-2 which bundles faults and attacks together into a single class. While the inter-classification due to similarities is a limitation to deep learning-based intrusion detection, the results obtained in this study are the best obtained so far for intrusion detection in the C-ITS for a wide array of misbehavior types.

In addition to the prediction results, run-time results for the deep learning models used in the three proposed DLCEs are shown in Table 9. We used two popular testbeds, namely, Nvidia Jetson Nano (Nano) and Raspberry Pi 3B (Rpi) for emulating the edge servers in this work. As shown in the table, the total time is split into data setup time (for converting the vehicle data into sequences) and the actual prediction time. The time taken is reported in milliseconds, with the data setup time being negligible in comparison with the actual prediction time. It can be observed that the *4-LSTM* and *2-CNN 1-LSTM* models take 288.07 ms and 251.56 ms respectively on Nano, and 570.48 ms and 337.16 ms respectively on Rpi. The runtime for DLCE-1 would be the runtime of *2-CNN 1-LSTM* model, whereas that of DLCE-2 and DLCE-3 would be the sum of the runtime of the deep learning models used in each of the stages. Thus, looking at the predictive and time-based performance of the classifiers, DLCE-1 is best suited as a complete misbehavior classification engine. When the time constraints are more relaxed, DLCE-3 which is the second-best proposed classification engine can also be deployed. Besides, the first stages of DLCE-2 and DLCE-3 can be used in situations where only coarse-grained classification is necessary, whereas their second stage can be used when the fine-grained classification of faults and attacks is needed. Thus the proposed DLCEs can suit a wide variety of situations.

7. Conclusion

This work aims to design a viable deep learning-based classification and intrusion detection scheme suited to next-generation C-ITS applications such as the Internet of Vehicles (IoV). It presents a deep learning-based intrusion detection scheme that can run on individual edge servers, deployed at the IoV network edge and co-located with the RSUs. Single-stage and two-stage classifiers are designed, which comprise of coarse-grained and fine-grained classification stages. Several deep learning models are trained on cloud servers in this experiment. These pre-trained models are run on testbeds emulating edge servers to detect and classify different vehicle misbehavior types from the test dataset. The results of these experiments show that a single-stage fine-grained classification performs the best, however, two-stage classification (comprising an initial coarse-grained classification into different classes followed by a fine-grained classification) also performs on par. Depending on the application scenario and the input vehicular traffic data available, one may choose to prioritize deploying one type of classifier over the other. Given the general nature of the proposed classifiers, it is possible to classify

similar misbehavior types with high prediction metrics. Future works can focus on extending the use of the proposed classifiers to more vehicular misbehavior types and their identified combinations. In addition, studies can also be conducted to classify misbehavior types based on the area of misbehavior such as time, position, velocity, acceleration, and ID, instead of the individual misbehavior type itself, thereby making a more general classification scheme applicable to a variety of vehicular behavior.

Acknowledgement

The work of Vinay Chamola and F. Richard Yu was supported in part by the SICI SICRG Grant through the Project Artificial Intelligence Enabled Security Provisioning and Vehicular Vision Innovations for Autonomous Vehicles, and in part by the Government of Canada's National Crime Prevention Strategy and Natural Sciences and Engineering Research Council of Canada (NSERC) CREATE Program for Building Trust in Connected and Autonomous Vehicles (TrustCAV).

References

[1] S. Garg, K. Kaur, G. Kaddoum, S. H. Ahmed, D. N. K. Jayakody, Sdn-based secure and privacy-preserving scheme for vehicular networks: A 5g perspective, IEEE Transactions on Vehicular Technology 68 (9) (2019) 8421–8434.

[2] C.-F. Cheng, G. Srivastava, J. C.-W. Lin, Y.-C. Lin, Fault-tolerance mechanisms for software-defined internet of vehicles, IEEE Transactions on Intelligent Transportation Systems 22 (6) (2021) 3859–3868.

[3] R. Khan, P. Kumar, D. N. K. Jayakody, M. Liyanage, A survey on security and privacy of 5g technologies: Potential solutions, recent advancements, and future directions, IEEE Communications Surveys & Tutorials 22 (1) (2019) 196–248.

[4] Z. Lv, L. Qiao, J. Li, H. Song, Deep-learning-enabled security issues in the internet of things, IEEE Internet of Things Journal 8 (12) (2020) 9531–9538.

[5] T. Alladi, V. Chamola, N. Kumar, et al., Parth: A two-stage lightweight mutual authentication protocol for uav surveillance networks, Computer Communications 160 (2020) 81–90.

[6] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021) 619–640.

[7] Z. Lu, G. Qu, Z. Liu, A survey on recent advances in vehicular network security, trust, and privacy, IEEE Transactions on Intelligent Transportation Systems 20 (2) (2018) 760–776.

[8] R. Lu, L. Zhang, J. Ni, Y. Fang, 5g vehicle-to-everything services: Gearing up for security and privacy, Proceedings of the IEEE 108 (2) (2019) 373–389.

[9] T. Alladi, V. Chamola, R. M. Parizi, K.-K. R. Choo, Blockchain applications for industry 4.0 and industrial iot: A review, IEEE Access 7 (2019) 176935–176951.

[10] T. Alladi, S. Chakravarty, V. Chamola, M. Guizani, A lightweight authentication and attestation scheme for in-transit vehicles in iov scenario, IEEE Transactions on Vehicular Technology 69 (12) (2020) 14188–14197.

[11] M. Kamal, G. Srivastava, M. Tariq, Blockchain-based lightweight and secured v2v communication in the internet of vehicles, IEEE Transactions on Intelligent Transportation Systems.

[12] Z. Lv, L. Qiao, A. Kumar Singh, Q. Wang, Ai-empowered iot security for smart cities, ACM Transactions on Internet Technology 21 (4) (2021) 1–21.

[13] Z. Lv, L. Qiao, S. Verma, Ai-enabled iot-edge data analytics for connected living, ACM Transactions on Internet Technology 21 (4) (2021) 1–20.

[14] W. Wu, Y. Huang, R. Kurachi, G. Zeng, G. Xie, R. Li, K. Li, Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks, IEEE Access 6 (2018) 45233–45245.

[15] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, R. Poovendran, Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks, IEEE Transactions on Information Forensics and Security 14 (9) (2019) 2300–2314.

[16] B. Groza, P.-S. Murvay, Efficient intrusion detection with bloom filtering in controller area networks, IEEE Transactions on Information Forensics and Security 14 (4) (2018) 1037–1051.

[17] O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeny, E. M. Awwad, M. A. Elmeligy, M. A. Mohamed, H. Malik, An intelligent secured framework for cyberattack detection in electric vehicles' can bus using machine learning, IEEE Access 7 (2019) 127580–127592.

[18] K. Zhu, Z. Chen, Y. Peng, L. Zhang, Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using lstm, IEEE Transactions on Vehicular Technology 68 (5) (2019) 4275–4284.

[19] N. Moustafa, M. Keshk, E. Debie, H. Janicke, Federated ton_iot windows datasets for evaluating ai-based security applications, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2020, pp. 848–855.

[20] T. Alladi, V. Chamola, B. Sikdar, K.-K. R. Choo, Consumer iot: Security vulnerability case studies and solutions, IEEE Consumer Electronics Magazine 9 (2) (2020) 17–25.

[21] V. Chamola, C.-K. Tham, S. Gurunarayanan, N. Ansari, et al., An optimal delay aware task assignment scheme for wireless sdn networked edge cloudlets, Future Generation Computer Systems 102 (2020) 862–875.

[22] V. Chamola, A. Sancheti, S. Chakravarty, N. Kumar, M. Guizani, An iot and edge computing based smart parking operations framework for v2g system, IEEE Transactions on Vehicular Technology.

[23] G. S. S. Chalapathi, V. Chamola, A. Vaish, R. Buyya, Industrial internet of things (iiot) applications of edge and fog computing: A review and future directions, Fog/Edge Computing For Security, Privacy, and Applications (2021) 293–325.

[24] B. Hussain, Q. Du, S. Zhang, A. Imran, M. A. Imran, Mobile edge computing-based data-driven deep learning framework for anomaly detection, IEEE Access 7 (2019) 137656–137667.

[25] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, J.-P. Hubaux, Eviction of misbehaving and faulty nodes in vehicular networks, IEEE Journal on Selected Areas in Communications 25 (8) (2007) 1557–1568.

[26] N. Kumar, N. Chilamkurti, Collaborative trust aware intelligent intrusion detection in vanets, Computers & Electrical Engineering 40 (6) (2014) 1981–1996.

[27] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, M. Rajarajan, Host-based intrusion detection for vanets: a statistical approach to rogue node detection, IEEE transactions on vehicular technology 65 (8) (2015) 6703–6714.

[28] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo, X. Zeng, A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network, IEEE Access 7 (2019) 154560–154571.

[29] T. Zhang, Q. Zhu, Distributed privacy-preserving collaborative intrusion detection systems for vanets, IEEE Transactions on Signal and Information Processing over Networks 4 (1) (2018) 148–161.

[30] F. van Wyk, Y. Wang, A. Khojandi, N. Masoud, Real-time sensor anomaly detection and identification in automated ve-

hicles, IEEE Transactions on Intelligent Transportation Systems 21 (3) (2019) 1264–1276.

[31] L. Nie, Z. Ning, X. Wang, X. Hu, Y. Li, J. Cheng, Data-driven intrusion detection for intelligent internet of vehicles: A deep convolutional neural network-based method, IEEE Transactions on Network Science and Engineering.

[32] L. Nie, Y. Li, X. Kong, Spatio-temporal network traffic estimation and anomaly detection based on convolutional neural network in vehicular ad-hoc networks, IEEE Access 6 (2018) 40168–40176.

[33] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, D. Gan, Cloud-based cyber-physical intrusion detection for vehicles using deep learning, Ieee Access 6 (2017) 3491–3508.

[34] T. Alladi, V. Kohli, V. Chamola, F. R. Yu, Securing the internet of vehicles: A deep learning-based classification framework, IEEE Networking Letters 3 (2) (2021) 94–97.

[35] T. Alladi, V. Kohli, V. Chamola, F. R. Yu, M. Guizani, Artificial intelligence (ai)-empowered intrusion detection architecture for the internet of vehicles, IEEE Wireless Communications.

[36] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, R. Y. Kwok, Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme, IEEE Transactions on Cognitive Communications and Networking 5 (4) (2019) 1060–1072.

[37] C. Chen, J. Jiang, N. Lv, S. Li, An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge, IEEE Access 8 (2020) 99059–99069.

[38] R. Q. Hu, L. Hanzo, et al., Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks, IEEE Transactions on Vehicular Technology 68 (4) (2019) 3086–3099.

[39] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, Y. Zhang, Deep learning empowered task offloading for mobile edge computing in urban informatics, IEEE Internet of Things Journal 6 (5) (2019) 7635–7647.

[40] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, M. S. Hossain, Mobility-aware proactive edge caching for connected vehicles using federated learning, IEEE Transactions on Intelligent Transportation Systems (2020) 1–11.

[41] A. Dalgkitsis, P.-V. Mekikis, A. Antonopoulos, C. Verikoukis, Data driven service orchestration for vehicular networks, IEEE Transactions on Intelligent Transportation Systems (2020) 1–10.

[42] VeReMi Extension, `https://github.com/josephkamel/VeReMi-Dataset`, [Online; accessed 20-September-2020] (2020).

[43] J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien, F. Kargl, Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets, in: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, 2020, pp. 1–6.

[44] I. Mahmoudi, J. Kamel, I. Ben-Jemaa, A. Kaiser, P. Urien, Towards a reliable machine learning-based global misbehavior detection in c–its: Model evaluation approach, in: Vehicular Ad-hoc Networks for Smart Cities, Springer, 2020, pp. 73–86.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: