DeepADV: A Deep Neural Network Framework for Anomaly Detection in VANETs

Article *in* IEEE Transactions on Vehicular Technology · September 2021

DOI: 10.1109/TVT.2021.3113807

5 authors, including:

Tejasvi Alladi
Birla Institute of Technology and Science Pilani
24 PUBLICATIONS **2,152** CITATIONS
SEE PROFILE

Vinay Chamola
Birla Institute of Technology and Science Pilani
227 PUBLICATIONS **10,269** CITATIONS
SEE PROFILE

DeepADV: A Deep Neural Network Framework for Anomaly Detection in VANETs

Tejasvi Alladi, *Senior Member, IEEE*, Bhavya Gera, Ayush Agrawal, Vinay Chamola, *Senior Member, IEEE* and Fei Richard Yu, *Fellow, IEEE*

***Abstract*—We are seeing a growth in the number of connected vehicles in Vehicular Ad-hoc Networks (VANETs) to achieve the goal of Intelligent Transportation System (ITS). This is leading to a connected vehicular network scenario with vehicles continuously broadcasting data to other vehicles on the road and the roadside network infrastructure. The presence of a large number of communicating vehicles greatly increases the number and types of possible anomalies in the network. Existing works provide solutions addressing specific anomalies in the network only. However, since there can be a multitude of anomalies possible in the network, there is a need for better anomaly detection frameworks that can address this unprecedented scenario. In this paper, we propose an anomaly detection framework for VANETs based on deep neural networks (DNNs) using a sequence reconstruction and thresholding algorithm. In this framework, the DNN architectures are deployed on the roadside units (RSUs) which receive the broadcast vehicular data and run anomaly detection tasks to classify a particular message sequence as anomalous or genuine. Multiple DNN architectures are implemented in this experiment and their performance is compared using key evaluation metrics. Performance comparison of the proposed framework is also drawn against the prior work in this area. Our best performing deep learning-based scheme detects anomalous sequences with an accuracy of 98%, a great improvement over the set benchmark.**

***Index Terms*—Vehicular ad-hoc networks (VANETs), Deep Neural Networks (DNNs), Intelligent Transportation System (ITS), anomaly detection, thresholding, security.**

I. INTRODUCTION

Vehicular Ad-hoc Networks (VANETs) can assist in achieving the goal of Intelligent Transportation Systems (ITSs) by supporting applications such as blind-spot warnings, traffic violation detection, and other road safety applications. With the unprecedented growth of connected vehicles in this scenario, a large amount of vehicular traffic data is expected to be shared in VANETs. Thus, along with providing several benefits, this opens up several pathways for attackers to target the security of these networks [1–3].

The nodes may exchange malicious information leading to various types of anomalies in the network and thus affecting the entire network [4, 5]. Any form of data manipulation in the sensor data generated and the shared messages transmitted by the vehicles in the network can have serious repercussions including fatal accidents. In this regard, several methods have been proposed for improving security in vehicular networks [6, 7]. However, it is a serious challenge to secure these networks with several types of anomalies possible, e.g., Data Replay attacks target the integrity of the network; while Denial of Service (DoS) and Disruptive attacks target the availability of the network. The heterogeneous nature of the attacks tends to create unavoidable loopholes in traditional security methods making it extremely hard to prevent all types of attacks and detect all types of anomalies in VANETs. Therefore, it becomes quintessential to have counteractive measures to strengthen the security of the network.

The evolution of deep learning techniques reveals the tremendous potential in securing the vehicular networks [8, 9]. Most of the past anomaly detection schemes for vehicular networks address one or more types of anomalies in the network. However, given the heterogeneous nature of anomalies in the network, there is a need for a robust anomaly detection scheme to detect any type of anomaly in the network. With the growing number of anomaly types, it is ideal to design a framework that learns the genuine vehicular data pattern in the network. And any deviation from this learned pattern can be flagged as anomalous. Even with the lack of prior knowledge of the possible anomalies in the network, learning algorithms provide the flexibility to adapt to situations by recognizing patterns and extracting features. The availability of large amounts of data from vehicles pertaining to speed, position coordinates, identities, etc further incentivizes the use of learning algorithms [10–12].

In this paper, we propose an anomaly detection framework for securing VANETs using Deep Neural Network (DNN) architectures. The major contributions of this paper are highlighted below:

i. We designed DeepADV, a DNN-based anomaly detection framework for the identification of anomalous data pertaining to various possible anomalies in VANETs.
ii. Three types of anomaly scenarios, namely, fault only, attack only, and all anomaly (both faults and attacks)

Manuscript received January 10, 2021; revised June 18 and August 15, 2021; accepted September 15, 2021. This work was partially supported by the SICI SICRG grant received by Dr. Vinay Chamola and Prof. Richard Yu for the project Artificial Intelligence Enabled Security Provisioning and Vehicular Vision innovations for Autonomous Vehicles, and also through 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 (CAVs). The review of this article was coordinated by Dr. Hongzhi Guo (*Corresponding author: Vinay Chamola*).

T. Alladi and F. Richard Yu are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada. (e-mail: talladi.carleton@gmail.com and richard.yu@carleton.ca).

V. Chamola is with the Department of Electrical and Electronics Engineering & APPCAIR, BITS-Pilani, Pilani Campus, 333031, India. (e-mail: vinay.chamola@pilani.bits-pilani.ac.in).

B. Gera and A. Agrawal are with the Department of Computer Science and Information Systems, BITS-Pilani, Pilani Campus, 333031, India. (e-mail: f20170783@pilani.bits-pilani.ac.in and f20170599@pilani.bits-pilani.ac.in).

scenarios are considered for analyzing the performance of the proposed framework.

iii. Performance analysis of the framework using key evaluation metrics is presented by employing several DNN architectures.

iv. A visual representation of the reconstructed sequences and a follow-up discussion is also presented.

The rest of this paper is organized as follows. A discussion on the literature survey on anomaly detection and intrusion detection in VANETs is presented in Section II. A brief preliminary background on DNN architectures used in this paper is given in Section III. Section IV presents the network scenario and dataset preprocessing. The proposed anomaly detection framework is presented in Section V. The results for running anomaly detection on all the three scenarios in terms of computed evaluation metrics are presented in Section VI, and the performance analysis of the results is presented in Section VII. The paper is then concluded in Section VIII.

II. Literature Survey

Several approaches are available in the existing literature to solve anomaly detection problems in vehicular networks. Zaidi *et al.* [17] proposed a statistical approach for intrusion detection to identify rogue vehicles in the network. Garg *et al.* [18] incorporated Elliptic Curve Cryptography and Fuzzy C-means clustering for anomaly detection. Cho *et al.* [19] proposed an anomaly-based intrusion detection system with Recursive Least Squares algorithm and Cumulative Sum to detect any abnormalities in the vehicular traffic. Another work by the same authors [20] implement a novel detection scheme by measuring and utilizing voltages on the in-vehicle network to identify the attackers in the in-vehicular network. Xun *et al.* [21] presented an intrusion detection system to monitor message transmissions in real-time based on vehicle voltage signals in the controller area network bus. However, most of these schemes make use of traditional intrusion detection approaches.

We further discuss some machine learning-based detection schemes proposed in the literature. Tang *et al.* [22] provided a research survey on various machine learning techniques with applications in networking, communication, and security features in 6G vehicular networks. In another survey, the authors of [23] discussed various challenges of conventional technologies and machine learning-based solutions in vehicular networks. Gao *et al.* [24] proposed a machine learning-based intrusion detection framework that uses random forests. Zhang *et al.* [25] proposed a distributed scheme that can be implemented on individual vehicles in the network which can collaborate for anomaly information exchange. Another real-time authentication scheme is implemented in [26] using support vector domain description and Convolutional Neural Networks (CNNs). Choi *et al.* [27] leveraged electrical controller area network signals as fingerprints of the electronic control units for an automotive intrusion detection system to distinguish between errors and the bus-off attacks. Nie *et al.* [16] presented a method using the features of network traffic to implement network traffic estimation based on an anomaly detection algorithm.

Owing to the large amount of vehicular traffic data being generated, deep learning schemes may be better suited for next-generation VANET scenarios. Gui *et al.* and Kato *et al.* [28, 29] mention that deep learning-based strategies consist of complex neural network architectures that effectively capture dynamic scenarios and are potential solutions for attack prediction. These architectures also ensure good scalability, thus, using a comprehensive dataset adds to the efficiency of the deep learning-based solutions for attack detection in a complex network scenario [29]. In this regard, we further discuss deep learning-based detection frameworks proposed in the literature. Nie *et al.* [13] proposed an approach that uses spatio-temporal and sparse features of VANET traffic and a CNN architecture with a Mahalanobis distance-based loss function. Another work by the authors of *et al.* [15] proposed an architecture based on CNNs which resulted in better performance than Support Vector Machines and Principal Component Analysis methods. Kamel *et al.* [14] proposed a detection strategy to integrate the confidence range of sensors in the detection checks included in the messages transmitted in the network. Van *et al.* [30] also employed CNNs for detecting malicious data being transmitted by vehicles through sensors. Ning *et al.* [31] applied deep reinforcement learning over edge servers to further solve the computation issues in the intrusion detection problem. However, most of these works focus on one or two types of anomalies only, and thus they are focused on specific cases only. A comparison of some of these state-of-the-art works is presented in Table I.

In a popular survey on vehicular network security, Lu *et al.* [4] highlighted the major security issues in VANET scenarios. As discussed above, existing works provide solutions addressing specific anomalies in the network only. However, since there can be a multitude of anomalies possible, there is a need for better anomaly detection frameworks that can address several anomalies using a single integrated solution. In this work, we focus on the detection of most of these issues such as Denial of Service (DoS) attacks, data replay attacks, and Sybil attacks along with other attacks and faults encompassing several dimensions of network security such as availability, authenticity, and integrity as outlined in [4]. Realizing the need to develop a framework that is both cost-efficient and is also effective against several possible anomaly types, we propose a DNN-based framework that would help to serve this purpose. Kamel *et al*. in [32] proposed a comprehensive dataset and employed a consistency and plausibility check-based detection algorithm for anomaly detection. We use the dataset proposed in [32] and compare our results using their results as the benchmark. A discussion on this dataset is also provided in Section IV-B.

III. Preliminary Background

In this section, we give a brief background on the different types of DNNs employed in this paper. We exploit several deep learning architectures to implement our proposed scheme. Deep learning architectures consist of non-linear activation functions that could be helpful in the processing and learning of complex sequence patterns. We use efficient components

TABLE I: Comparison between various state-of-the-art anomaly detection schemes

Scheme	Methodology	Merit	Demerit
Nie *et al.* [13]	Uses spatio-temporal and sparse features of VANET traffic, and CNN architecture with a Mahalanobis distance based loss function	Based on the simulation results, the proposed architecture improves the True Positive Rate	Results by other models and loss functions have not been compared
Kamel *et al.* [14]	Integrated the confidence range of sensors in the detection checks which is included in the standard V2X messages	Confidence range field had a more reliable and fine-tuned detection capability in the V2X network	Comparison with other deep learning models on VeReMi dataset is not shown. Some attack types have very low precision
Nie *et al.* [15]	Proposed architecture based on CNN and a fundamental error term for the convergence of the back propagation algorithm	Compared with Neural Network, SVM and PCA methods shows better performance	Results are not compared with other deep learning models
Nie *et al.* [16]	CNN-based anomaly detection approach is used with hierarchical convolution and a threshold-based method for detection	Results show the effectiveness of the proposed method against other machine learning models	Comparison only with some machine learning models and not with any deep learning models

such as convolutional layers and LSTMs owing to their effective pattern capturing algorithms. In [13, 15, 16], authors compared deep learning algorithms such as CNNs against traditional machine learning algorithms such as PCA, the results consistently show markedly better performance for deep learning schemes. Hence we chose several deep learning architectures consisting of various important components that could support our proposed scheme.

1) Convolutional Neural Networks: Convolutional Neural Networks (CNNs) architecture consists of an input layer, multiple hidden layers, and an output layer. There are four main operations in CNNs, namely, convolution, non-linearity, pooling, and classification. These four operations are the basic building blocks of every CNN. They have the advantage of not requiring manual engineering to input features in sequence classification. Hence it provides better prospects of learning an internal representation of the time-series data.

2) Long Short Term Memory: The Long Short-Term Memory (LSTM) architecture was developed since the existing Recurrent Neural Networks (RNNs) had difficulty accessing the long time lags because of vanishing gradients and exploding gradients. Both of these problems were solved by LSTMs. An LSTM layer comprises recurrently connected blocks (memory blocks). Every memory block contains recurrently connected memory cells and three multiplicative units – the input, output, and forget gates. The input gate decides the information entering the cell state. The forget gate controls the degree to which the information remains in the cell and the output gate controls the magnitude to which the value in the cell determines the output activation of the LSTM unit.

3) Gated Recurrent Unit: GRU architecture is similar to LSTMs but lacks an output gate. GRU's ability to solve the vanishing gradient problem is by its use of an update gate and a reset gate. The update and reset gate control the information that flows into and out of the memory respectively. The main advantages of GRUs are in improving the memory capacity of an RNN and are simpler in computation and implementation as compared to LSTMs.

4) Attention: Attention vectors help in predicting the target through the relatedness of elements. The main advantage of using the attention mechanism is that the neural network can refer to the previous input sequence which otherwise would have to be encoded in one fixed-size vector.

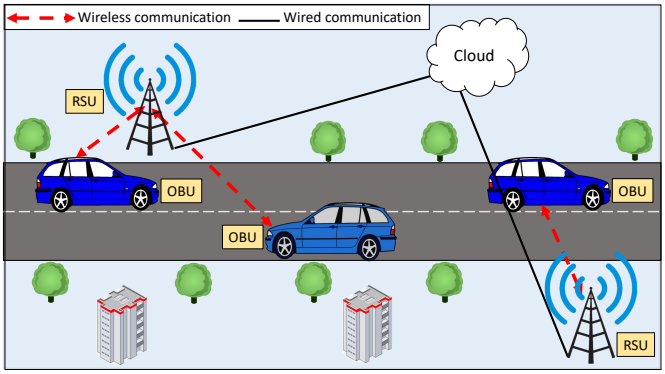


Fig. 1: The network scenario considered in this paper.

IV. Network Scenario and Data Pre-processing

In this section, we first discuss the network scenario considered in this paper. This is followed by a discussion on the dataset used and data preprocessing carried out on it.

A. Network Scenario

The VANET scenario considered in this paper has vehicles moving on the roads and broadcasting information in the network, as shown in Fig. 1. This information primarily consists of position and speed coordinates. This information sent from the vehicle On-board Units (OBUs) is received by the Road Side Units (RSUs), converted into sequences, and then reconstructed. The reconstruction error calculated is used later for the classification of the sequences into genuine or anomaly classes. In the figure, vehicles are labeled as OBUs. It is assumed that the DNNs are trained on the cloud server, and dedicated wired communication links exist between the cloud server and the RSUs. The vehicle OBUs and the RSUs communicate wirelessly, and the sequence reconstruction and prediction tasks run on the RSUs.

We use DNN architectures to approximate the sequence reconstructor. DNN models provide sophisticated functions and sufficient non-linearity which serve as universal function approximators. We use an unsupervised setting to make our architectures reconstruct the genuine sequences.

B. Data Preprocessing

We used the dataset provided by Kamel et al. [32, 33] for evaluating the proposed framework. Table II lists the details

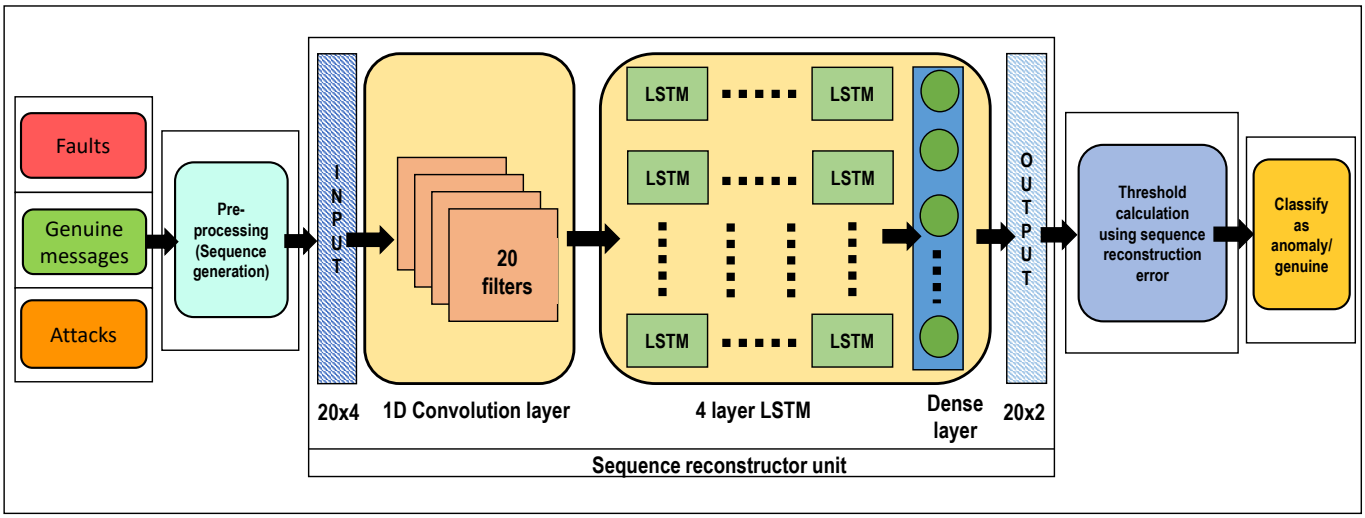


Fig. 2: Proposed anomaly detection framework (DeepADV).

of the various anomalies present in the dataset used in our experiment. The dataset is composed of various types of fault and attack type messages which are together considered as Anomaly class in this paper. The genuine vehicular messages are counted in the Genuine class.

The data fields present in the dataset are the send time, sender ID, sender pseudo ID, message ID, and position-based information - position, speed, acceleration, and heading. We used only X, Y coordinates of position and speed among the other data fields for the creation of our customized dataset. Multiple time sequences from the original dataset were created containing the X, Y coordinates of position and speed. A sliding window of half the sequence length was chosen for maintaining continuity of data across the sequences for better learning by the DNN architectures.

We aimed to have low repetition and a large sample size for our customized dataset. Slide length becomes a determining factor here - choosing a smaller slide length will create more repetition and hence more data, while a larger slide length produces lower repetition and less data. We chose a sequence length of 20 and a slide length of 10 to lower the repetition and simultaneously have a larger number of training samples. Once all the time sequences of a vehicle were generated, they were added to their corresponding anomaly type (fault/attack) in their corresponding fault-wise/attack-wise dictionary in the customized dataset created. This dataset is then used in the proposed framework while testing the deep learning architectures.

V. Proposed Anomaly Detection Framework

In this section, we discuss DeepADV, the proposed framework that can effectively detect various faults and attacks in the considered network scenario as anomalous. This framework is composed of five major steps listed below. It is also depicted in Fig. 2 with the best performing CNN-LSTM model as its DNN architecture.

i. Extracting messages from the data exchanged in the network (containing faults, attacks, and genuine messages).
ii. Pre-processing the data and converting it into sequences.
iii. Passing these sequences into the sequence reconstructor unit (a DNN architecture).
iv. Exploiting the error between the original sequences and the reconstructed sequences to calculate the potential threshold using a thresholding algorithm.
v. Using the calculated potential threshold to classify the sequences as anomalous or genuine.

The sequence reconstructor is a function that takes a sequence (anomalous/genuine) as input and produces a sequence whose pattern and values should be identical to the genuine data sequence. We present three different scenarios to show the working of the proposed framework: i.) Fault only scenario, ii.) Attack only scenario, and iii.) All anomaly (attacks and faults) scenario.

A. Architectures Employed

In the proposed framework, we employ six different DNN architectures - Model 1: CNN-LSTM (M-1), Model 2: CNN-bidirectional LSTM (M-2), Model 3: LSTM with attention (M-3), Model 4: Auto-encoders (M-4), Model 5: Stacked LSTM (M-5), and Model 6: GRUs (M-6). We considered these architectures taking into account their effectiveness in learning the internal representation of the sequence data and their competence in internal storage capacity. The configurations of their architectures are explained as follows. In M-1 and M-5, we used four layers of 256 LSTM units. In M-2 we used four layers of 256 bidirectional LSTM units. We used a single convolutional layer (1-D) in M-1 and M-2 with 20 filters each. In M-4, we used two encoder and decoder layers each with 256 and 128 units respectively. In M-3, we used an LSTM bidirectional layer of 32 units as a pre-attention layer and a simple LSTM layer of 64 units as a post-attention layer. The units for the LSTM and GRU models and the filters for the CNN models were finalized based on several experiments performed to arrive at the best results.

TABLE II: Details of anomaly and genuine types considered.

Type	Class	Description
0	Genuine	Genuine behavior
1	Anomaly (Fault)	Constant position
2	Anomaly (Fault)	Constant position offset
3	Anomaly (Fault)	Random position
4	Anomaly (Fault)	Random position offset
5	Anomaly (Fault)	Constant speed
6	Anomaly (Fault)	Constant speed offset
7	Anomaly (Fault)	Random speed
8	Anomaly (Fault)	Random speed offset
9	Anomaly (Fault)	Delayed messages
10	Anomaly (Attack)	Disruptive
11	Anomaly (Attack)	Data replay
12	Anomaly (Attack)	Eventual stop
13	Anomaly (Attack)	DoS
14	Anomaly (Attack)	DoS random
15	Anomaly (Attack)	DoS disruptive
16	Anomaly (Attack)	Data replay sybil
17	Anomaly (Attack)	Traffic congestion sybil
18	Anomaly (Attack)	DoS random sybil
19	Anomaly (Attack)	DoS disruptive sybil

B. Sequence Reconstruction

X_j is defined as the j^{th} sequence vector of the pre-processed data where $X_j = [(p_{x_1}, p_{y_1}, s_{x_1}, s_{y_1}), ..., (p_{x_{20}}, p_{y_{20}}, s_{x_{20}}, s_{y_{20}})]$ and X'_j is defined as the reconstructed sequence vector from j^{th} input sequence vector X_j, where $X'_j = [(p'_{x_1}, p'_{y_1}), .., (p'_{x_{20}}, p'_{y_{20}})]$. (p_x, p_y) are the position coordinates and (s_x, s_y) are the speed coordinates in the sequence vectors, and (p'_x, p'_y) are the reconstructed position coordinates. The model takes a 20×4 input in the form of position and speed coordinates and reproduces a 20×2 output in the form of position coordinates which are reconstructed by the sequence reconstructor.

C. Thresholding Algorithm for Anomaly Detection

To calculate the potential threshold, we define a precision value and the range of values starting from zero that could serve as thresholds. The Thresholding algorithm is shown in Algorithm 1. For each threshold in the range, incremented by the respective precision, the corresponding accuracy of classification is calculated. The threshold value corresponding to the maximum accuracy is taken as the potential threshold if the accuracy saturates after m number of iterations. At each iteration, precision is increased by a factor of 10 and the range gets limited to the region around the threshold value corresponding to the maximum accuracy. To calculate the accuracy corresponding to each threshold, the difference of Mean Absolute Error (MAE) between the input sequence and the reconstructed sequence, and the threshold value would decide whether the sequence is anomalous or not. A positive difference ($MAE > threshold$) would classify the sequence as anomalous while a negative difference ($MAE < threshold$) would classify the sequence as genuine. After m number of iterations, the threshold corresponding to the maximum accuracy is declared as the potential threshold to be used for classification.

Algorithm 1 Thresholding Algorithm

$N \leftarrow$ iterations corresponding to possible thresholds
$X \leftarrow$ test data sequences
$Y \leftarrow$ Anomaly type labels (1-19), 0 for genuine
$X' \leftarrow$ Reconstructed data sequences
$Accuracy[N] \leftarrow$ array of N elements
$threshold \leftarrow$ possible threshold
$st \leftarrow$ possible starting threshold
$pre \leftarrow$ precision of the possible thresholds
$m \leftarrow$ no. of iterations needed to reach the precision level
for each l in range(m) **do**
  for each i in range(N) **do**
    $Accuracy[i] \leftarrow 0$
    $Threshold[i] \leftarrow st + (i/pre)$
    for each j in range(len(X)) **do**
      $mae \leftarrow MAE(X[j], X'[j])$
      if ($mae < threshold$ **and** Y[j]=0) **or** ($mae > threshold[i]$ **and** Y[j]!=0) **then**
        $Accuracy[i] \leftarrow Accuracy[i] + (1/len(X))$
      end if
    end for
  end for
  $threshold \leftarrow st + (argmax(Accuracy))/pre)$
  $en \leftarrow st + ((argmax(Accuracy) + 1)/pre)$
  $st \leftarrow st + ((argmax(Accuracy) - 1)/pre)$
  $pre \leftarrow pre \times 10$
  $N \leftarrow (en - st) \times pre$
end for

An anomaly (fault or attack) would change the sequence pattern/values of the genuine sequence. We attempt to exploit this deviation of the anomalous behavior from the genuine behavior to effectively classify the sequences as anomalous or genuine. This potential threshold would serve as the benchmark for this classification.

D. Training and Testing

To make the models learn the trend and values of the genuine data sequences, we trained the six models only on the genuine data sequences. The training was carried out on Google Colaboratory cloud servers, which emulate the Cloud in our VANET model. Around 85% of the genuine data from the customized dataset is used for training. We used MAE as the loss function for backpropagation as well. We used the trained models as sequence reconstructors in three test scenarios: 1) Fault only scenario (Anomaly type 1-9), 2) Attack only scenario (Anomaly type 10-19), and 3) All anomaly scenario (Anomaly type 1-19) in combination with the genuine

sequences. The testing phase for predicting a particular vehicle type from the sampled sequences is carried out on Nvidia Jetson Nano, an embedded board that emulates the RSUs in our network. In each scenario, we randomly sampled around 15000 sequences of the genuine and attack/fault classes in the ratios identical to the original data set distribution and reconstructed the sequences. After reconstruction, we applied the thresholding algorithm to calculate the potential threshold with the corresponding maximum accuracy.

VI. Simulation Results

In this section, we discuss the test results for all the DNN architectures in terms of key evaluation metrics. For each scenario, we calculated the Precision (Pre), Recall (Rec), Accuracy (Acc), and F1 score (F1) for both the classes considered: Genuine class and Anomalous class. For each anomaly class, we also calculated the respective recall to gain insights about how accurately the framework can detect the particular anomaly class as anomalous. We first plot accuracy vs threshold graphs for all the considered scenarios in Fig. 3. In all three scenarios, we can see that model M-1 has the highest accuracy, followed by model M-2.

A. Fault Only Scenario

As discussed above, there might be a scenario where the faults are the major anomalies and need to be detected effectively. Hence, the proposed anomaly detection framework could take the form of a fault detection framework for this scenario. Table III shows the recall values for each of the 9 fault types (anomaly type 1 to type 9) considered along with the genuine class (type 0). And Table IV shows the overall precision, recall, and F1-score of Genuine and Anomaly (Fault) classes along with their accuracy of prediction. From these tables, it is evident that M-1 fits the proposed framework with an accuracy of 0.984 as compared to the other five architectures. Its computed precision is 0.982 for genuine and 0.996 for faults. The overall recall of fault and genuine types is 0.911 and 0.999 respectively. The F1 score of the architecture is 0.991 for genuine and 0.952 for fault type. M-6 shows the least performance with an overall accuracy of 0.957. Its precision is 0.954 and 0.997; recall is 0.999 and 0.766; and F1-score is 0.976 and 0.866 for genuine and fault types respectively.

M-1 detects 7 out of the 9 faults considered with more than 0.99 recall. All the other models can detect only 4 to 5 out of the 9 faults with more than 0.99 recall. While M-1 can detect fault type 2 with 0.835 recall, M-2 can moderately detect fault type 2 with 0.602 recall, the other four models are poorly detecting fault type 2 with less than 0.11 recall. M-1 and M-2 are detecting fault type 9 with 0.374 and 0.439 recalls respectively. The other four models are poorly detecting fault type 9 with less than 0.11 recall. M-1, M-2, M-3, and M-5 can perfectly detect fault types 3 to 8 with a recall of 1.0.

TABLE III: Recalls for fault only scenario

	DNN Architecture					
Type	**M-1**	**M-2**	**M-3**	**M-4**	**M-5**	**M-6**
0	0.999	0.997	0.999	0.998	0.996	0.999
1	0.993	0.981	0.940	0.951	0.925	0.914
2	0.835	0.602	0.0	0.102	0.019	0.004
3	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	0.988
6	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	0.938	1.0	1.0
9	0.374	0.439	0.0	0.103	0.061	0.0

TABLE IV: Evaluation metrics for fault only scenario

Model	**Class**	**Prec**	**Rec**	**F1**	**Acc**
M-1	Genuine	0.982	0.999	0.991	0.984
	Anomaly	0.996	0.911	0.952	
M-2	Genuine	0.978	0.997	0.988	0.979
	Anomaly	0.986	0.891	0.936	
M-3	Genuine	0.955	0.999	0.977	0.961
	Anomaly	0.998	0.770	0.869	
M-4	Genuine	0.958	0.998	0.978	0.962
	Anomaly	0.989	0.787	0.877	
M-5	Genuine	0.956	0.996	0.976	0.959
	Anomaly	0.978	0.777	0.866	
M-6	Genuine	0.954	0.999	0.976	0.957
	Anomaly	0.997	0.766	0.866	

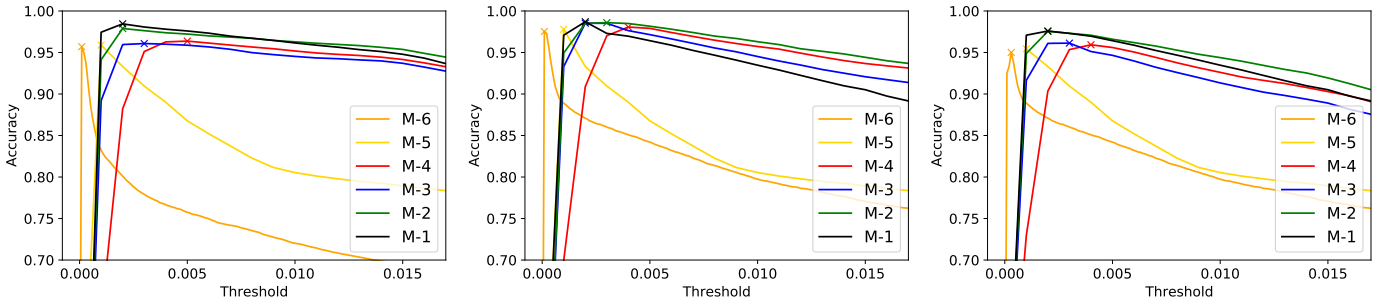
B. Attack Only Scenario

Similar to the fault-only scenario, there could also be a situation where the attacks are in majority and a separate detection system for attacks might be an effective solution. Table V show the recalls of the 10 attack types considered (anomaly type 10 to type 19) along with the genuine class (type 0). Table VI shows the evaluated overall precision, recall, and F1-score of the Genuine and Anomaly (attack) classes with their prediction accuracy. The model M-1 performs the best with 0.987 accuracy followed by M-2 and M-3 with 0.986 accuracy. For M-1, precision values are 0.982 and 0.998; recall values are 0.999 and 0.964; and F1-scores are 0.990 and 0.981 for the genuine and attack types respectively. M-6 has the least accuracy of 0.976 among all the models.

Both M-1 and M-6 can detect 8 out of 10 attacks with more than 0.99 recall. While M-6 poorly detects attack type 12 with a recall of 0.158, other models can moderately detect it with a recall of around 0.6. M-5 is detecting attack type 13 with a recall of 0.755 while the other models are detecting the same with a recall of at least 0.85. Attack types 10, 14, 18, and 19 are detected with a recall of 1.0 by M-1. In addition to these attacks, M-6 can detect attack types 11 and 17 also with a recall of 1.0.

C. All Anomaly Scenario

In addition to the fault only and attack only scenarios, we also present a scenario in which both of them can be present as anomalies. Table VII shows the recalls of all the 19 anomalies and Table VIII shows the precision, recall, and F1 scores of the Genuine and Anomaly classes along with their accuracy of detection. Consistent with the earlier scenarios, M-1 performs



(a) Fault only scenario (b) Attack only scenario (c) All anomaly scenario

Fig. 3: Accuracy vs Threshold graphs for all the three scenarios.

TABLE V: Recalls for attack only scenario

	DNN Architecture					
Type	**M-1**	**M-2**	**M-3**	**M-4**	**M-5**	**M-6**
0	0.999	0.999	1.0	0.993	0.997	0.994
10	1.0	0.981	0.995	0.995	0.972	1.0
11	0.995	1.0	0.995	0.991	0.991	1.0
12	0.627	0.598	0.627	0.641	0.632	0.158
13	0.883	0.868	0.858	0.853	0.755	0.865
14	1.0	1.0	1.0	1.0	1.0	1.0
15	0.999	0.998	0.999	0.994	0.993	0.997
16	0.993	0.993	0.985	0.970	0.988	0.998
17	0.995	0.986	1.0	0.991	1.0	1.0
18	1.0	1.0	1.0	1.0	1.0	1.0
19	1.0	1.0	1.0	0.998	1.0	1.0

TABLE VI: Evaluation metrics for attack only scenario

Model	**Class**	**Prec**	**Rec**	**F1**	**Acc**
M-1	Genuine	0.982	0.999	0.990	0.987
	Anomaly	0.998	0.964	0.981	
M-2	Genuine	0.979	0.999	0.989	0.986
	Anomaly	0.998	0.960	0.979	
M-3	Genuine	0.979	1.0	0.989	0.986
	Anomaly	1.0	0.959	0.979	
M-4	Genuine	0.977	0.993	0.985	0.980
	Anomaly	0.986	0.954	0.970	
M-5	Genuine	0.971	0.996	0.984	0.978
	Anomaly	0.993	0.943	0.967	
M-6	Genuine	0.971	0.994	0.982	0.976
	Anomaly	0.989	0.942	0.965	

the best with an accuracy of 0.98; a precision of 0.969 and 0.996; a recall of 0.997 and 0.956; and an F1-score of 0.983 and 0.976 for the Genuine and Anomaly classes respectively. M-6 performs with the least accuracy of 0.95.

Both M-1 and M-6 can detect 14 out of 19 anomalies with more than 0.99 recall. All the anomalies' recalls are consistent with the previous two scenarios considered. While M-1 performs moderately for anomalies 2, 9, and 12, M-6 performs poorly in these.

VII. Analysis and Discussion

In this section, we analyze the performance of the proposed framework based on the prediction results obtained. We then

TABLE VII: Recalls for all anomaly scenario

	DNN Architecture					
Type	**M-1**	**M-2**	**M-3**	**M-4**	**M-5**	**M-6**
0	0.997	0.996	0.998	1.0	0.996	0.993
1	0.978	0.963	0.947	0.866	0.914	0.925
2	0.892	0.645	0.0	0.0	0.011	0.027
3	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	0.994	1.0
6	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0
9	0.519	0.432	0.011	0.0	0.055	0.027
10	0.995	0.989	1.0	1.0	0.984	1.0
11	0.995	0.995	0.995	0.995	1.0	1.0
12	0.636	0.565	0.658	0.625	0.625	0.152
13	0.887	0.866	0.866	0.847	0.753	0.832
14	1.0	1.0	1.0	1.0	1.0	1.0
15	0.995	0.998	0.997	0.998	0.990	0.997
16	0.999	0.999	0.986	0.998	0.984	0.994
17	1.0	1.0	1.0	0.989	0.989	1.0
18	1.0	1.0	1.0	1.0	1.0	1.0
19	1.0	1.0	1.0	1.0	0.997	1.0

TABLE VIII: Evaluation metrics for all anomaly scenario

Model	**Class**	**Prec**	**Rec**	**F1**	**Acc**
M-1	Genuine	0.969	0.997	0.983	0.980
	Anomaly	0.996	0.956	0.976	
M-2	Genuine	0.959	0.996	0.977	0.973
	Anomaly	0.995	0.941	0.967	
M-3	Genuine	0.938	0.998	0.967	0.960
	Anomaly	0.997	0.908	0.950	
M-4	Genuine	0.938	0.995	0.965	0.958
	Anomaly	0.992	0.908	0.948	
M-5	Genuine	0.929	0.996	0.961	0.953
	Anomaly	0.994	0.894	0.941	
M-6	Genuine	0.926	0.993	0.959	0.950
	Anomaly	0.989	0.890	0.937	

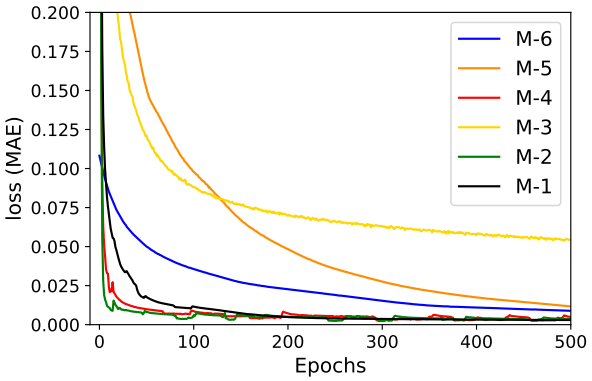


Fig. 4: Learning curves.

compare the different DNN architectures to be chosen for the framework and discuss the anomalies where the performance is moderate. We also visualize the reconstruction plots for the reconstructed sequences to get better insights into the performance analysis.

A. Comparative Analysis of the Considered Models

From the results in the previous section, it is clear that CNN-LSTM (M-1) performs the best in all the scenarios as compared to the other five models. Similarly, GRU (M-6) performs with the least accuracy for all the scenarios. While CNN-LSTM bidirectional (M-2) is the second-best performer in terms of accuracy, LSTM-attention (M-3) and Auto-encoders (M-4) both perform similarly in all the scenarios, followed by M-2. In every scenario, stacked LSTM (M-5) performs a little better compared to the GRU (M-6) model. Fig. 4 shows the learning curves for all the models considered. It can be seen that M-1 and M-2 converge faster and end up at lower loss (MAE) compared to other models.

Convolutional layers in the CNN-based architectures can capture the spatio-temporal features from the sequences. This adds robustness to the framework by enabling the sequence reconstructor to learn complex patterns in the sequences pertaining to the Genuine class. Therefore, both the CNN-LSTM and CNN-LSTM bidirectional models can detect some anomaly types such as type 2 and type 9 moderately which are poorly detected by the models. Also, all the LSTM based models performed better than the GRU model which is evident from the anomaly type 12 detection.

B. Analyzing Moderately Detected Anomalies

Results show that for a few fault/attack types, the proposed framework is performing with less than 0.8 recall. Anomaly type 2 is the Constant Position Offset fault. Since the position coordinates in this fault are being added/subtracted by a constant offset, it is tough to detect. However, the proposed framework comprising of the CNN-LSTM model can detect this anomaly type with 0.892 recall since it utilizes the preprocessing memorizing capability of CNNs and LSTMs and uses deviation as a measure to detect anomalies. Anomaly type 9 is the Delayed Messages fault. Sequences pertaining to this anomaly type have genuine vehicle information except that they are sent at a delay of some time unit making the values and the trend of the data identical to the genuine sequences. Therefore, the framework poorly detects this type with 0.519 recall. Similarly, anomaly type 12 is moderately detected with a 0.636 recall. Anomaly type 12 is the Eventual Stop attack which results in null speed values and frozen position coordinates. For some sequences, the values, as well as the trend, might be similar to that of genuine data making it tough for the framework to detect. All the recall values mentioned here are concerning all anomaly scenario employing CNN-LSTM architecture. This analysis is consistent with the other two scenarios as well.

TABLE IX: Real-time results (in milliseconds) for prediction on Nvidia Jetson Nano

	Model			
Time	**M-1**	**M-2**	**M-5**	**M-6**
Data setup	235.83	370.92	222.43	224.94
Prediction	0.299	0.395	0.339	0.33
Total	236.13	371.32	222.77	225.27

TABLE X: Comparison with prior work

	Evaluation Metrics			
Scheme	**Pre**	**Rec**	**F1**	**Acc**
DeepADV	0.996	0.956	0.976	0.980
[32]	0.991	0.823	0.899	0.929

C. Real-time Prediction Results

Here we show the real-time performance of the proposed framework in terms of the time taken for prediction of a vehicle's time sequence data on *Nvidia Jetson Nano*, which emulates the RSUs in the considered VANET network. Table IX shows the prediction time results of two best performing models (in terms of evaluation metrics), M-1 & M-2, and two worst-performing models (in terms of evaluation metrics), M-5 & M-6. As can be seen from the table, M-1 takes considerably less time compared to M-2, whereas it performs almost on part with M-5 and M-6.

D. Reconstruction Plots

We further show the reconstruction plots of the sequences concerning the CNN-LSTM architecture which is the best performing model for the proposed framework. Visualization of the original and the reconstructed sequences could give us direction to interpret the results and relate to the framework's performance. Fig. 5 shows the reconstruction plots of genuine sequences, sequences with all fault types, and sequences with all attack types. In Fig. 6, we show the reconstruction plots with two anomaly types that were perfectly detected as anomalous (anomaly (fault) type 3 and anomaly (attack) type 19) and the plots of those types with poor detection performance (anomaly (fault) type 9 and anomaly (attack) type 12). From the reconstruction plots, it is evident that the larger

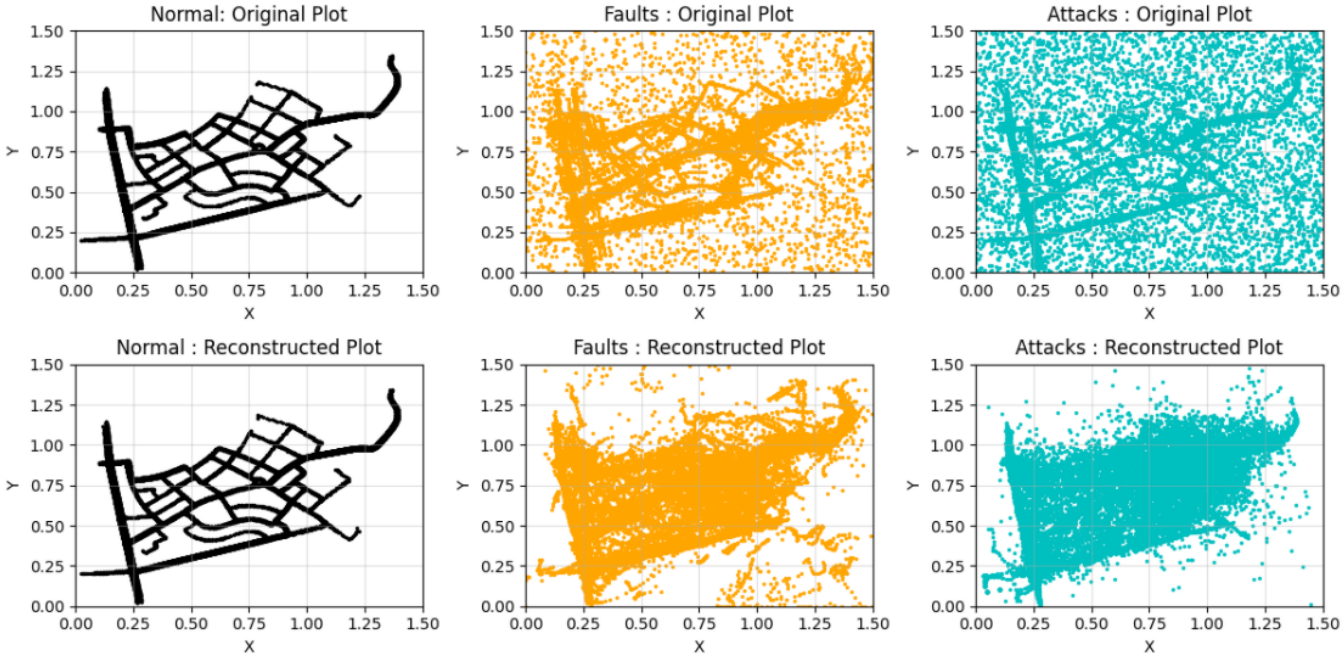


Fig. 5: Reconstruction of genuine (left), all faults (middle) and all attacks (right).

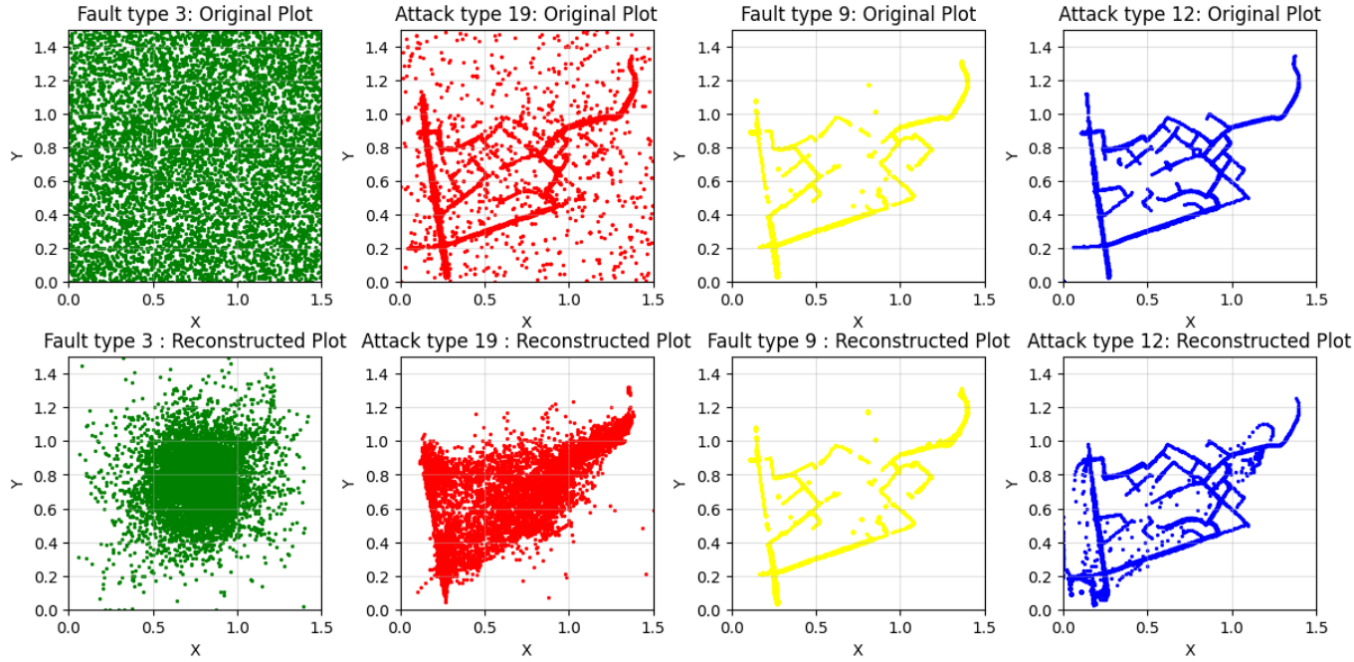


Fig. 6: Reconstruction of the best and the worst performing anomalies.

the divergence of the input anomalous sequence from the genuine sequences, the larger would be the MAE value of the reconstructed sequence with the input, resulting in its accurate predicted class. The thresholding algorithm thus becomes an important aspect of the overall framework for deciding the classification bar.

E. Comparison With Prior Works

Several recent works [13, 15, 16, 32] proposed schemes for effective anomaly detection. Authors in [13, 16] worked on DDOS attacks and their scheme detected anomalies with the recall in the range of 0.9 to 0.95. In Table X, we compare our results with the work presented in [32] for the all anomaly scenario. Our proposed scheme achieved greater performance with respect to all the metrics. Moreover, the proposed schemes in these papers do not take into account various anomaly types and different architectures for deep learning schemes. We take various anomaly types (19 types) and provide a comparison with various deep learning architectures in various scenarios. Our proposed scheme uses only the

genuine sequence data to train the architecture for effective sequence pattern recognition. Our scheme also promises to classify an alien attack/fault type considering that it can detect 19 anomalies using a single threshold with 0.987 accuracy. Since the architecture is trained only on the genuine sequence data, the same trained architecture could be used for identifying any other type of anomaly based on its reconstruction of the genuine sequences and the previously calculated threshold. Moreover, after verifying the other anomalies that were not taken into consideration, a new potential threshold could also be calculated using the Thresholding Algorithm 1 without retraining the model. However, as described in previous sections, it is important to note that an anomaly type should have a deviated pattern in terms of occurrence of values as compared with the genuine sequence pattern to detect it with maximum recall.

VIII. Conclusion

With the rise in the number of connected vehicles in the VANET scenario to achieve the goal of Intelligent Transportation Systems (ITSs), there is an unprecedented growth of vehicular traffic data. This leads to an increase in the number and type of anomalies in the communicated data. In this paper, DeepADV, a robust DNN-based anomaly detection framework was proposed taking unknown anomalies also into consideration. The proposed framework is based on the reconstruction error calculation. The performance of the framework is analyzed based on six different DNN models considered. Recalls for several anomaly types are calculated corresponding to each DNN model. And for classification into genuine and anomaly classes, the models are compared against key evaluation metrics. Reconstruction plots for several anomaly types are also presented to show the effectiveness of the proposed framework. The best-performing model of this framework is also shown to perform better in comparison to the existing framework.

References

[1] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo, "Consumer iot: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[2] Y. Li, Q. Luo, J. Liu, H. Guo, and N. Kato, "Tsp security in intelligent and connected vehicles: Challenges and solutions," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 125–131, 2019.

[3] T. Alladi, A. Agrawal, B. Gera, V. Chamola, B. Sikdar, and M. Guizani, "Deep neural networks for securing iot enabled vehicular ad-hoc networks," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[4] Z. Lu, G. Qu, and Z. Liu, "A survey on recent advances in vehicular network security, trust, and privacy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 760–776, 2018.

[5] R. Lu, L. Zhang, J. Ni, and Y. Fang, "5g vehicle-to-everything services: Gearing up for security and privacy," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 373–389, 2019.

[6] G. Bansal, N. Naren, V. Chamola, B. Sikdar, N. Kumar, and M. Guizani, "Lightweight mutual authentication protocol for v2g using physical unclonable function," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7234–7246, 2020.

[7] T. Alladi, S. Chakravarty, V. Chamola, and M. Guizani, "A lightweight authentication and attestation scheme for in-transit vehicles in iov scenario," *IEEE Transactions on Vehicular Technology*, 2020.

[8] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[9] J. Zhang, H. Guo, and J. Liu, "Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1736–1745, 2020.

[10] P. Chhikara, R. Tekchandani, N. Kumar, V. Chamola, and M. Guizani, "Dcnn-ga: a deep neural net architecture for navigation of uav in indoor environment," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4448–4460, 2020.

[11] H. Grover, T. Alladi, V. Chamola, D. Singh, and K.-K. R. Choo, "Edge computing and deep learning enabled secure multi-tier network for internet of vehicles," *IEEE Internet of Things Journal*, 2021.

[12] T. Alladi, V. Kohli, V. Chamola, F. R. Yu, and M. Guizani, "Artificial intelligence (ai)-empowered intrusion detection architecture for the internet of vehicles," *IEEE Wireless Communications*, vol. 28, no. 3, pp. 144–149, 2021.

[13] L. Nie, H. Wang, S. Gong, Z. Ning, M. S. Obaidat, and K.-F. Hsiao, "Anomaly detection based on spatio-temporal and sparse features of network traffic in vanets," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[14] J. Kamel, "Misbehavior detection for cooperative intelligent transport systems (c-its)," Ph.D. dissertation, Institut polytechnique de Paris, 2020.

[15] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng, and Y. Li, "Data-driven intrusion detection for intelligent internet of vehicles: A deep convolutional neural network-based method," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2219–2230, 2020.

[16] L. Nie, Y. Li, and X. Kong, "Spatio-temporal network traffic estimation and anomaly detection based on convolutional neural network in vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 40 168–40 176, 2018.

[17] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for vanets: a statistical approach to rogue node detection," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6703–6714, 2015.

[18] S. Garg, K. Kaur, G. Kaddoum, S. H. Ahmed, and D. N. K. Jayakody, "Sdn-based secure and privacy-preserving scheme for vehicular networks: A 5g perspective," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8421–8434, 2019.

[19] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 911–927.

[20] ——, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1109–1123.

[21] Y. Xun, Y. Zhao, and J. Liu, "Vehicleeids: A novel external intrusion detection system based on vehicle voltage signals," *IEEE Internet of Things Journal*, 2021.

[22] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network toward 6g: Machine-learning approaches," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 292–307, 2019.

[23] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[24] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo, and X. Zeng, "A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network," *IEEE Access*, vol. 7, pp. 154 560–154 571, 2019.

[25] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for vanets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 148–161, 2018.

[26] Y. Xun, J. Liu, N. Kato, Y. Fang, and Y. Zhang, "Automobile driver fingerprinting: A new machine learning based authentication scheme," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1417–1426, 2019.

[27] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[28] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6g: Opening new horizons for integration of comfort, security, and intelligence," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 126–132, 2020.

[29] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6g," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96–103, 2020.

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

Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1264–1276, 2019.

[31] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, and R. Y. Kwok, "Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, 2019.

[32] J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien, and 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.

[33] "VeReMi Extension," https://github.com/josephkamel/VeReMi-Dataset, 2020, [Online; accessed 20-September-2020].

Tejasvi Alladi obtained his Ph.D. degree from the Birla Institute of Technology and Science (BITS), Pilani, India in 2021. He obtained his B.E. and M.S. degrees from BITS, Pilani and the North Carolina State University, USA in 2010 and 2015 respectively. He is currently working as a Postdoctoral Researcher in the Department of Systems and Computer Engineering, Carleton University, Canada. He also has around 6 years of industrial experience working on embedded systems in MNCs such as Qualcomm Technologies and Samsung Electronics. His research interests include developing security solutions for the Internet of Things using cryptography, deep learning, and blockchain technologies.

Bhavya Gera is pursuing B.E.(Hons.) Computer Science and MSc.(Hons.) Economics at Birla Institute of Technology and Science (BITS), Pilani. She is currently a thesis student at Heidelberg University. She is a WISE Virtuell and DAAD-WISE recipient, a prestigious scholarship to work on Computer Science research in Germany and has been an attendee of the prestigious Indian Workshop on Applied Deep Learning. She has also won various maths olympiads and received NTSE and Grace Hopper Scholarship. Her research interests and work include Federated Learning, Financial Engineering, NLP and Deep Learning for Security Provisioning.

Ayush Agrawal is pursuing B.E. (Hons.) Computer Science and M.Sc. (Hons.) Economics at Birla Institute of Technology and Science (BITS), Pilani. He is currently working as a research intern at Microsoft Research India in Automated Theorem Proving and Game Playing using Reinforcement Learning. He is a recipient of DAAD-WISE (Germany) and MITACS-GRI (Canada) scholarship awards along with IASc-SRFP and RBI (India) fellowships. He has been working on various applied fields in Deep Learning such as Medical Imaging, Music Information Retrieval, IoT Security and Computer Vision with various international collaborations. His research interests include Econometrics and Artificial Intelligence in general.

Vinay Chamola received the B.E. degree in electrical and electronics engineering and master's degree in communication engineering from the Birla Institute of Technology and Science, Pilani, India, in 2010 and 2013, respectively. He received his Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2016. In 2015, he was a Visiting Researcher with the Autonomous Networks Research Group (ANRG), University of Southern California, Los Angeles, CA, USA. He also worked as a post-doctoral research fellow at the National University of Singapore, Singapore. He is currently Assistant Professor with the Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani where he heads the Internet of Things Research Group / Lab. He has over 75 publications in high ranked SCI Journals including more than 55 *IEEE Transaction*, Journal and Magazine articles. His research interests include IoT Security, Blockchain, UAVs, VANETs, 5G and Healthcare. He serves as an Area Editor for the Ad Hoc Networks journal, Elsevier. He also serves as an Associate editor in the IEEE Internet of Things Magazine, IEEE Networking letters, IET Quantum Communications, IET Networks and several other journals. He is a Guest Editor in Computer Communication, Elsevier; and also the IET Intelligent Transportation Systems Journal. He serves as co-chair of various reputed workshops like in IEEE Globecom Workshop 2021, IEEE ANTS 2021, IEEE ICIAfS 2021 to name a few. He is co-founder and President of a healthcare startup Medsupervision pvt. ltd. He is a senior member of the IEEE.

Fei Richard Yu received the PhD degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a startup (in San Diego, CA, USA), where he worked on the research and development in the areas of advanced wireless communication technologies and new standards. He joined Carleton School of Information Technology and the Department of Systems and Computer Engineering (cross-appointment) at Carleton University, Ottawa, in 2007, where he is currently a Professor. His research interests include cyber-security, connected and autonomous vehicles, artificial intelligence, blockchain, and wireless systems. He has published 600+ papers in reputable journals/conferences, 8 books, and 28 granted patents, with 10,000+ citations (Google Scholar). He received the IEEE TCGCC Best Journal Paper Award in 2019, Distinguished Service Awards in 2019 and 2016, Outstanding Leadership Award in 2013, Carleton Research Achievement Awards in 2012 and 2021, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009 and the Best Paper Awards at IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, IEEE/IFIP TrustCom 2009 and Int'l Conference on Networking 2005. He serves on the editorial boards of several journals, including Co-Editor-in-Chief for Ad Hoc Sensor Wireless Networks, Lead Series Editor for IEEE Transactions on Vehicular Technology, IEEE Communications Surveys Tutorials, and IEEE Transactions on Green Communications and Networking. He has served as the Technical Program Committee (TPC) Co-Chair of numerous conferences. He is a "Highly Cited Researcher" according to Web of Science since 2019. He is an IEEE Distinguished Lecturer of both Vehicular Technology Society (VTS) and Comm. Society. He is an elected member of the Board of Governors of the IEEE VTS and Editor-in-Chief for IEEE VTS Mobile World newsletter. He is a Fellow of the IEEE, Canadian Academy of Engineering (CAE), Engineering Institute of Canada (EIC), and Institution of Engineering and Technology (IET).