Deep Neural Networks for Securing IoT Enabled Vehicular Ad-Hoc Networks

Tejasvi Alladi[1], Ayush Agrawal[2], Bhavya Gera[2], Vinay Chamola[1], *Senior Member, IEEE*, Biplab Sikdar[3], *Senior Member, IEEE*, Mohsen Guizani[4], *Fellow, IEEE*
[1]Department of Electrical and Electronics Engineering, BITS Pilani, Pilani Campus, India
[2]Department of Computer Science and Engineering, BITS Pilani, Pilani Campus, India
[3]Department of ECE, National University of Singapore, Singapore
[4]CSE Department, Qatar University, Qatar

***Abstract*—Vehicular ad-hoc network (VANET) security has been an active area of research over the past decade. However, with the increasing adoption of the Internet of Things (IoT) in VANETs, the number of connected vehicles is set to grow exponentially over the next few years, which translates to a higher number of communication interfaces and a greater possibility of cybersecurity attacks. Along with these cybersecurity attacks, the instances of compromised vehicles sending faulty information about their positions and speeds also increase exponentially. Thus, there is a need to augment the existing security schemes with anomaly detection schemes which can differentiate normal vehicle data from malicious and faulty data. Since, the number of anomaly types can be many, deep neural networks would work best in this scenario. In this paper, we propose a deep neural network-based vehicle anomaly detection scheme. We use a sequence reconstruction approach to differentiate normal vehicle data from anomalous data. Numerical results show that we can correctly detect data corresponding to several anomaly types.**

***Index Terms*—Vehicular ad-hoc networks (VANETs), Internet of Things (IoT), deep neural networks, deep learning, reconstruction, security.**

I. Introduction

Vehicular ad-hoc networks (VANETs) are important components of smart city environments due to their ability to improve safety, quality of life, and security of vehicle users. With the adoption of IoT, they can cater to applications ranging from blind spot warnings to traffic violation detection. Each node in a VANET has an On-Board Unit (OBU), with the nodes communicating among each other, and with the Road Side Units (RSUs).

Any node exchanging malicious information compromises the entire network. The sensor data and shared messages transmitted by the vehicle OBUs in VANETs must be trustworthy since any data tampering could lead to major mishaps including fatal accidents. Owing to their data reliance, VANETs involve the generation and exchange of large amounts of data shared through wireless communication. This opens several pathways for attackers to target the security of these networks [1]. Several security schemes have been proposed for vehicular networks that provide confidentiality of the messages exchanged [2, 3]. However, attacks such as data replay target the integrity of the network, while attacks such as Denial of Service (DoS) and disruptive attack target the availability of the network. Due to the wide variety of attacks, and the unavoidable loopholes in traditional security methods, it is not possible to entirely prevent attacks on VANETs. Therefore, it is imperative to have counteractive measures in addition to preventive measures to strengthen the security of the network.

The rise of artificial intelligence, machine learning, and deep learning has shown great potential in securing different types of IoT networks by meeting the security requirements of confidentiality, integrity, and availability [4]. In VANETs, the data collected from the vehicles such as vehicle kinematics (vehicle speed and position information), vehicle identities, and several other types of data may be used to train vehicle detection models to identify anomalous behavior in the system. Due to the lack of prior knowledge of the attack/fault type, learning algorithms provide the flexibility to adapt to situations by recognizing patterns and extracting features. The availability of a large amount of data further incentivizes the use of learning algorithms, with special emphasis on deep learning due to their significant increase in performance in such scenarios [5, 6].

In this paper, we propose an anomaly detection scheme for securing IoT enabled VANETs using deep neural networks. The major contributions of this paper are highlighted below:

i. A deep neural network-based anomalous vehicle detection scheme in VANETs deployed on the RSUs.
ii. Classification of normal vehicle data and anomalous vehicle data in the network.
iii. Identification of anomalous data pertaining to various attacks/faults.

The rest of this paper is organized as follows. Section II discusses the related works in securing vehicular networks using machine learning and deep learning techniques. Section III presents the network model, the dataset used in our paper, and the dataset preprocessing involved. Section IV presents our proposed deep neural network-based anomaly detection scheme. We present the testing results and discussion in Section V and conclude the paper in Section VI.

II. Related Work

In light of the flexibility that the learning algorithms offer as compared to traditional methods [7], we discuss the machine learning and deep learning approaches existing in the literature for securing VANETs.

Gao et al. [8] proposed an intrusion detection scheme using a classification algorithm based on Random Forests. Zhang et al. proposed a distributed detection scheme that can run on individual vehicles in the network and can collaborate for anomaly information exchange [9]. Yet another work by Garg et al. [10] proposed a machine learning-based intrusion detection scheme incorporating Elliptic Curve Cryptography and Fuzzy C-means clustering. However, machine learning approaches may not be sufficient with the growing amounts of data in IoT enabled VANETs.

Neural networks have also been experimented with for achieving the security requirements of VANETs. Van et al. exploited CNNs for detecting anomalous sensor data being transmitted by vehicles [11], however, only a few anomalous types were considered in their approach. Nie et al. [12, 13] also proposed a couple of intrusion detection approaches in recent works. In [12], an unsupervised approach using CNNs was explored which was extended further using a supervised technique in their latest work [13]. Even in these works, only a limited number of anomaly types were considered. A cloud-based intrusion detection scheme was proposed by Loukas et. al. [14]. This work can be considered as a solution for limited computation resources on the vehicle OBUs. However, it is still inefficient to offload deep learning tasks onto the remote cloud. Hence, there is a requirement to locally run these models closer to the vehicle nodes.

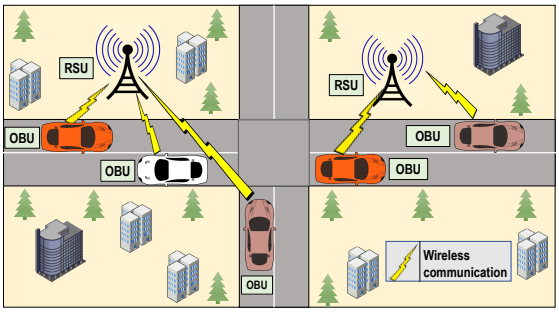

Fig. 1: Network model.

Addressing the above-discussed issues arising out of the processing requirements of the available vehicle data, and the need for running the deep learning tasks locally, we propose our neural network architecture in this paper. The models in this architecture run on the nearby RSUs with which the vehicles passing on the roads communicate. Further, the architecture is designed considering the growing number of anomalies in VANETs.

III. Network Model

The network model in this paper considers a vehicular network consisting of vehicle OBUs and RSUs. This network model is presented in Fig. 1. The communication between the vehicle OBUs and the RSUs can happen either using cellular V2X (CV2X) or Dedicated Short-Range Communication (DSRC) wireless protocols. The data broadcast by the vehicle OBUs is received by the RSUs. The RSUs pass the input data through the proposed anomaly detection scheme for possible anomaly detection in the network.

In this network, the vehicles are vulnerable to several different types of attacks such as DoS attack, data replay attack, sybil attack, and disruptive attack [15]. Along with these attack types, vehicles may also transmit incorrect information in the network (such as faulty position or speed coordinates) either inadvertently or intentionally. Our proposed scheme considers all these attacks and fault types. We discuss the dataset used and its preprocessing steps related to the proposed scheme below.

A. Dataset

In our paper, we used the *VeReMi Extension* dataset [16], which considers 19 different attack and fault types and 1 normal vehicle data type. Together we consider all 19 attack and fault types as an anomaly class and the normal type as a normal class. The dataset comprises the messages broadcast by the vehicle OBUs and received by the RSUs. Each such message/data point consists of the sending timestamp, X and Y position coordinates, and its differentials such as velocity, acceleration, and heading. The two X, Y coordinate values each of position and speed were used in our study.

Multiple time sequences each of these 20 data points were created from the original vehicle data, where each sequence corresponds to 20 vehicle messages transmitted from a vehicle OBU. These (20x4) sequences were generated with a slide length of 10, i.e., the next sequence is 10 data points after the start of the previous sequence, creating an overlap of 10 data points between one sequence and the next. This overlap leads to the repetition of some data points and thus to more data. Such sequences were created for each of the vehicle data types considered, i.e. for all 20 types. However, only the normal data types were used for training the models, while the sequences created for the rest of 19 anomaly types were used as part of the testing dataset.

IV. Proposed Anomaly Detection Scheme

We propose a sequence reconstruction based anomaly detection scheme using deep neural networks with the capability of detecting anomalies in the vehicular data. Any deviation from normal vehicle data is considered an anomaly in this scheme. The architecture represents a function that takes in time sequences generated from the position and speed coordinates of the normal vehicle data and reconstructs the position coordinates through a complex network of nonlinear functions.

A. Thresholding algorithm

We employ an unsupervised setting to detect the anomalous sequences. While the architecture is trained only on the normal data, we apply an algorithm to calculate the potential threshold to classify the sequences as normal or anomalous.

Since the architecture is trained only on the normal data, we expect that it has learned the pattern and the value range of the normal sequences.

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 normal
$X' \leftarrow$ Reconstructed data sequences
$Accuracy[N] \leftarrow$ array of N elements
$Threshold[N] \leftarrow$ array of N elements
$st \leftarrow$ possible starting threshold
$pre \leftarrow$ precision of the possible thresholds
for each i in range(N) **do**
    $Threshold[i] \leftarrow st + (i/pre)$
    for each j in range(len(X)) **do**
        $mae \leftarrow MAE(X[j], X'[j])$
        if ($mae < Threshold[i]$ **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

Thus, it should be able to reconstruct the normal position coordinates. If anomalous data (irrespective of its type) is given as input to this architecture, the combination of trained weights and nonlinear functions would force the architecture to generate the position coordinates whose pattern and values should be similar to the pattern and values of normal sequence data. Using the Mean Absolute Error (MAE) of the input sequence and the reconstructed sequence, we calculate the threshold above which the sequence will be classified as anomalous as shown in Algorithm 1. For calculating the potential threshold, we plot a graph between threshold and accuracy. The algorithm is repeatedly run by deciding the new st and pre variables until we get a sharp peak in the graph. The threshold corresponding to the maximum accuracy shown by the sharp peak would serve as a benchmark to classify normal and anomalous data.

B. Reconstruction

We define the input sequence of the architecture, $X_j = [(p_{x1}, p_{y1}, s_{x1}, s_{y1}), ..., (p_{x20}, p_{y20}, s_{x20}, s_{x20})]$ and the reconstructed output sequence, $X'_j = [(p'_{x1}, p'_{y1}), .., (p'_{x20}, p'_{y20})]$. Here, X_j is the j^{th} sequence from the input data, (p_x, p_y) are the position coordinates and (s_x, s_y) are the

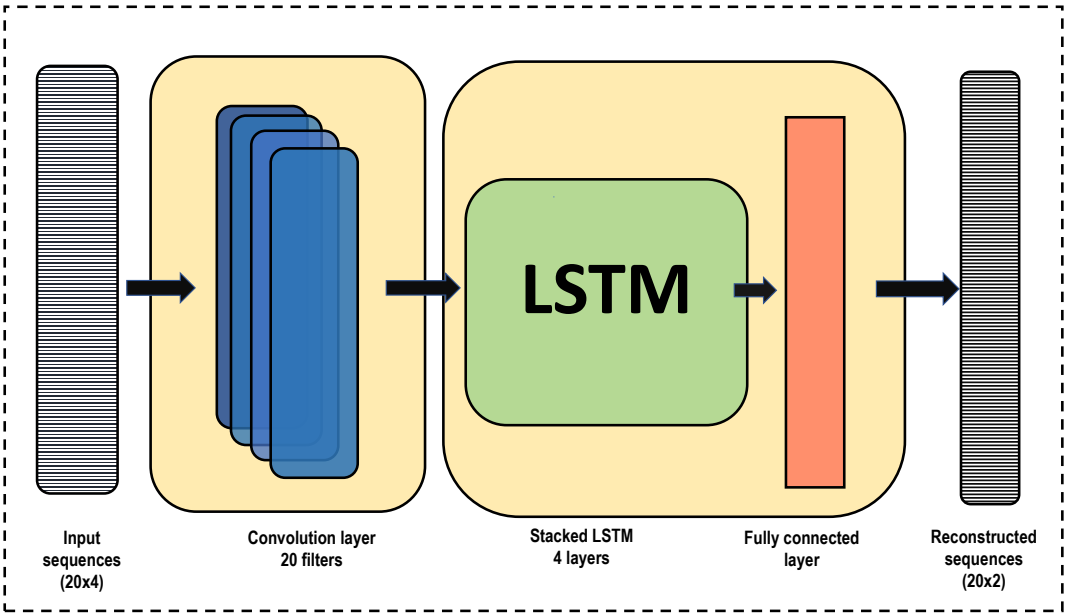

Fig. 2: Proposed CNN-LSTM architecture for anomaly detection.

speed coordinates, while X'_j is the corresponding reconstructed sequence, and (p'_x, p'_y) are the reconstructed speed coordinates. Thus, the model is taking a (20x4) input in the form of position and speed coordinates with a sequence length of 20 and reproducing a (20x2) output in the form of position coordinates which are supposed to be reconstructed through the trained pipeline and are similar to the pattern/values of the normal data sequence.

C. Training and Testing

We considered two popular neural network architectures for the implementation of the deep neural network models in this scheme. CNNs have the advantage of not requiring manual engineering for feature extraction in sequence classification. Hence, they provide better prospects of learning the internal representation of the time-series data. Long Short-Term Memory (LSTM) offers the advantage of being insensitivity to gap length and information storage capability from the previous time intervals.

Using CNNs and LSTMs, we implemented two deep neural network models: CNN-LSTM (a 1-D layer Convolution with 4-layer LSTM) and a stacked 4-layer LSTM. The CNN-LSTM model uses a single-dimensional convolutional layer with 20 filters to process the input data sequence and further passes the processed sequences into the stacked LSTM model with 4 layers of 256 units each to reconstruct the input sequence. CNN captures the spatio-temporal features that may be useful for similar (normal) pattern reconstruction. Features extracted by the CNN layer which are fed to LSTMs in the form of time-series data could add to the functionality of the LSTMs in detecting various attacks/faults affecting the messages sent in the form of position or speed distortions. We compare the CNN-LSTM model with the 4-layer stacked LSTM having 256 units. The proposed CNN-LSTM architecture to be employed in the anomaly detection scheme is shown in Fig. 2.

Since this is an unsupervised setting, our models are trained only on normal data sequences. 85% of the normal data is used for training. The testing data comprises of the normal and 1-19 anomalous data (1-9: faults, 10-19: attacks) in the ratios identical to the original data set distribution. MAE of the input sequence and the output sequence is used as a loss function to back-propagate the errors during training. After training the models, we used the thresholding algorithm to calculate the potential threshold for each model. Using the thresholds, we found the accuracy, precision, recall, and F1-scores. For all the anomalies (1-19), we calculated the recalls indicating how accurately our model can detect a particular anomaly type as an actual anomaly. We then compare our models based on these results.

TABLE I: Computed recalls of anomalies (1-19) for Model1: CNN-LSTM and Model2: Stacked LSTM

Model	Rec																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Model1	0.978	0.892	1.0	1.0	1.0	1.0	1.0	1.0	0.519	0.995	0.995	0.636	0.887	1.0	0.995	0.998	1.0	1.0	1.0
Model2	0.914	0.011	1.0	1.0	0.994	1.0	1.0	1.0	0.055	0.984	1.0	0.625	0.753	1.0	0.990	0.984	0.989	1.0	0.997

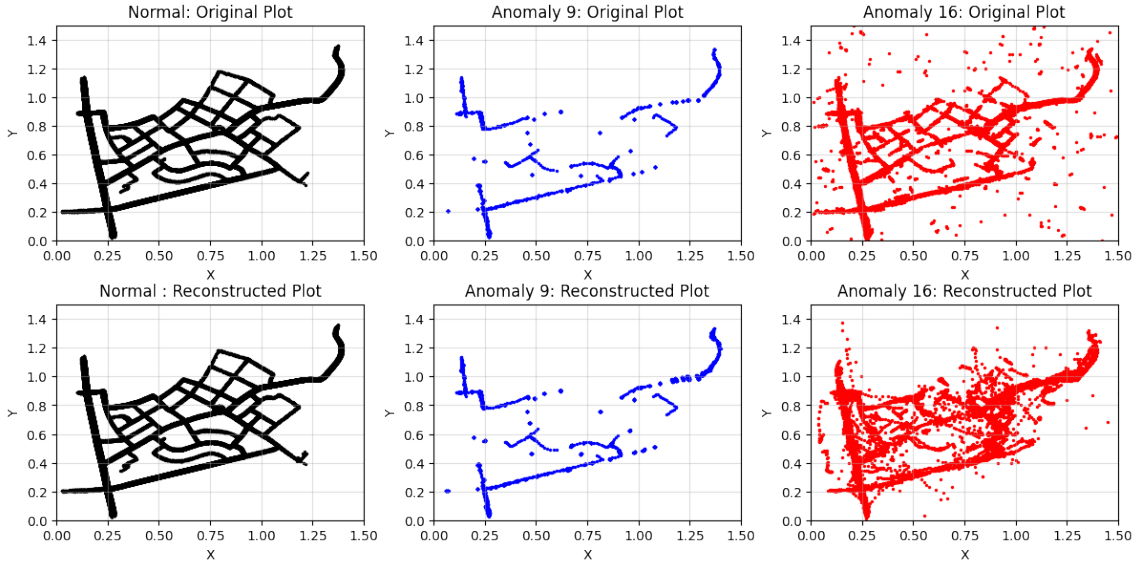

Fig. 3: Reconstruction of normal (left), anomaly type 9 (middle) and anomaly type 16 (right).

TABLE II: Overall evaluation metrics for Model1: CNN-LSTM and Model2: Stacked LSTM

Model	Class	Prec	Rec	F1	Acc
Model1	Normal	0.970	0.997	0.983	0.980
	Anomaly	0.996	0.956	0.976	
Model2	Normal	0.930	0.996	0.962	0.954
	Anomaly	0.995	0.894	0.942	

V. Results and Discussion

In Table I, we show the recalls calculated for each of the 19 anomaly types. The evaluation metrics precision (Prec), recall (Rec), F1 score (F1), and accuracy (Acc) for both the models on the test dataset are presented in Table II. From both the tables, we can infer that the proposed CNN-LSTM model (accuracy: 0.98) outperforms the stacked-LSTM model (accuracy: 0.954). Moreover, the computed precision, recall, and F1-scores of both the classes indicate the superiority of the CNN-LSTM model over the LSTM model.

A. Analysing the various anomalies

Anomaly type 2 is the constant offset added to the position coordinates. While normal stacked LSTM is poorly detecting this fault type (recall of 0.11), CNN-LSTM can detect it with a recall of 0.892. This anomaly type becomes difficult to detect since the pattern of the sequence remains the same as normal vehicle data, but all the values of the coordinates are increased/decreased with an offset. CNN layers capture the spatio-temporal features from the sequences which might contribute to the recognition of this anomaly. Anomaly type 9 is the eventual stop that freezes the position coordinates and sets the speeds to null values. This is poorly detected by both the models, with a recall of 0.55 in stacked LSTM and 0.519 in the CNN-LSTM model, since some sequences may have a similar trend as the normal data. Anomaly type 12 is the delayed messages attack in which higher overload might cause a delay in sending messages. Both the models perform moderately with the recall of 0.636 (CNN-LSTMs) and 0.625 (LSTMs). Anomaly type 13 (DoS attack) is detected with a recall of 0.887 in CNN-LSTM and 0.753 in stacked LSTM. For all the other anomaly types, both the models performed well with recalls of above 0.90. In every scenario, CNN-LSTM outperformed stacked LSTM with the overall recall of 0.956 against the 0.894 recall of stacked LSTM for 1-19 anomalies. Moreover, CNN-LSTM detected 14 out of 19 anomalies with

more than 0.99 recall. This shows the enhanced capability brought by the CNN layer pre-processing in the model. This also shows the importance of the thresholding algorithm which enables the model that is simply trained on normal data to detect various faults and attacks effectively.

B. Reconstruction visualizations

Fig. 3 shows the reconstruction of the normal data sequences, anomaly type 9, and anomaly type 16 by the proposed CNN-LSTM model. Normal data sequences are perfectly reconstructed with significantly less MAE, while the model is not able to reconstruct anomaly type 16 (data replay) data sequence because of the deviation of the input sequence pattern from normal data sequences resulting in higher MAE and thus making it easier to classify the sequences as anomalous. However, for attack type 9 as discussed in the analysis above, our model can reconstruct the anomalous sequences partially, resulting in a low recall of 0.519. From the reconstruction graphs, it is evident that the greater the deviation, the greater is the detection performance. Our proposed model is thus capable of reconstructing only the normal data sequence. Any deviation from the pattern/values would result in larger MAE with respect to the threshold, thus, correctly classifying the sequence as anomalous.

VI. Conclusion

In this paper, we proposed a deep neural network architecture for securing IoT enabled VANETs. Deep learning models were trained on time sequences generated from normal vehicle data in the network. The time sequences in the test data were passed through the trained models to reconstruct the original sequences. Based on the reconstruction error obtained by using the thresholding algorithm, we were able to classify sequences into normal data and anomalies. Further, by calculating the recalls for each anomalous class, we were able to perfectly classify most of the attack/fault types as anomalous. Based on the performance metrics evaluated, the proposed CNN-LSTM model is shown to give a higher performance compared to a stacked LSTM model.

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] 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*, 2020.

[3] T. Alladi, V. Chamola, R. M. Parizi, and K.-K. R. Choo, "Blockchain applications for industry 4.0 and industrial iot: A review," *IEEE Access*, vol. 7, pp. 176 935–176 951, 2019.

[4] 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*, 2020.

[5] 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*, 2020.

[6] A. Mehra, M. Mandal, P. Narang, and V. Chamola, "Reviewnet: A fast and resource optimized network for enabling safe autonomous driving in hazy weather conditions," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[7] C. Chen, J. Jiang, N. Lv, and S. Li, "An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge," *IEEE Access*, vol. 8, pp. 99 059–99 069, 2020.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] L. Nie, Z. Ning, X. Wang, X. Hu, Y. Li, and 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*, 2020.

[14] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *Ieee Access*, vol. 6, pp. 3491–3508, 2017.

[15] 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.

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