A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security

Min-Ju Kang and Je-Won Kang
Department of Electronics Engineering, Ewha W. University
52 Ewha-Ro Seodaemun-Gu, Seoul, Republic of Korea
Email: joo2950@ewhain.net and jewonk@ewha.ac.kr

Abstract—**In this paper, we propose a novel intrusion detection technique using a deep neural network (DNN). In the proposed technique, in-vehicle network packets exchanged between electronic control units (ECU) are trained to extract low-dimensional features and used for discriminating normal and hacking packets. The features perform in high efficient and low complexity because they are generated directly from a bitstream over the network. The proposed technique monitors an exchanging packet in the vehicular network while the feature are trained off-line, and provides a real-time response to the attack with a significantly high detection ratio in our experiments.**

I. Introduction

Today's vehicular system embeds a number of computer devices called Electronic Control Unit (ECU) to control and monitor the subsystems [1]. ECU communicates each other over in-vehicle networks to facilitate advanced automotive applications such as a carputer and an auto-driving service. Controller Area Network (CAN) [2] provides a simple and reliable communication protocol as the de factor standard of a in-vehicle network, connecting not only sensors and controllers but also the Internet. The adoption of CAN accelerates the applications with the emergence of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication interfaces [3]. However, the openness of the vehicular system increases the risk of malicious cyber attacks that can severely damage human life. Protection mechanisms in the vehicular networks are concerned with new security threats.

However, the conventional in-vehicle networks are tremendously vulnerable with the cyber attacks as CAN is developed for an isolated physical system before. For example, every ECU sharing a CAN bus can obtain any ECU-to-ECU message. Furthermore, a CAN packet has no sender's identification as shown in Fig. 1. Recent research works point the weakness of the security. Koscher *et al.* [4] conduct several experiments where a CAN message can be readily fuzzed by a packet injection and modification. Various attack scenarios e.g. disabling brakes and displaying wrong information on an instrument panel are shown in [5], [6].

There are research works to tackle the intrusion to a secured system. Hoppe *et al.* propose an intrusion detection system (IDS) in a vehicle [7]. They identify notable attack patterns such as increasing message numbers and missing message IDs, which can be used for detecting attacks. Larson et al. develop a specification-based approach for attack detection [8]. Their method compares the behavior of the current specification system with the pre-defined patterns. However, a system can be limited to collect all the possible patterns. Muter *et al.* suggest sensor-based attack detection comprising a few attack detection sensors to recognize if an event is a normal behavior or a type of an attack [9]. They try to identify an attack based on a basic level of a signal, but the detection ratio is not significantly remarkable despite the costs of the sensors. Muter *el al.* show the applicability of entropy-based anomaly detection using information theory [10].

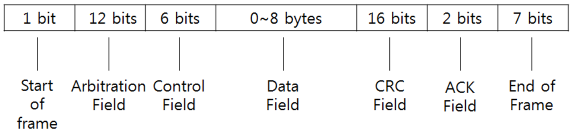


Fig. 1. Format of the CAN 2.0 data frame.

In this paper, we propose a novel intrusion detection approach exploiting a recent progress of deep neural network (DNN) [11], [12]. DNN has been extensively studied in a machine learning research field, and widely used for practical applications in computer vision and image processing, speech recognition, etc [12]. DNN is adopted to this paper as it shows remarkable classification performance [11]. Our proposed method trains high-dimensional CAN packet data to figure out the underlying statistical properties of normal and hacking CAN packets and extract the corresponding features. Once the features are trained off-line, the proposed system monitors an exchanging packet in a vehicular network to decide whether a system is being attacked, or not. The system provides an instant response to the attack as DNN requires little computations in the decision.

The rest of the paper is organized as follows. In Section II, we show the proposed method. Experimental results are shown in Section III. Finally, we conclude with remarks and show future works in Section IV.

II. Proposed Method

A. Overview of proposed intrusion detection system

The proposed intrusion detection system includes a monitoring module and a profiling module as in the conventional intrusion detection systems [13], as shown in Fig. 2. The monitoring module decides a type of an incoming CAN

978-1-5090-1698-3/16/$31.00 ©2016 IEEE

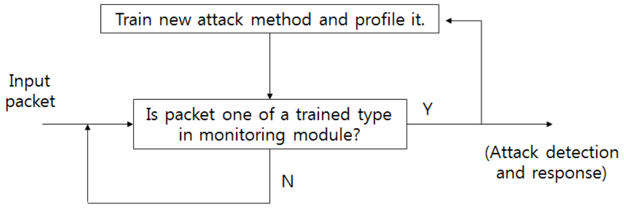

Train new attack method and profile it.
Input packet
Is packet one of a trained type in monitoring module?
Y
N
(Attack detection and response)

Fig. 2. A profiling module and a monitoring module in the proposed IDS.

packet based on trained features of known attacks. Once the monitoring module identifies a new attack, the profiling module records the attack model and update a system for an upcoming packet. The monitoring module and profiling module would be embedded in each ECU to analyzing a CAN packet. Detailed explanations on mechanisms of the modules will be given in sequel.

B. CAN packet feature

The proposed method extracts a CAN packet feature as an abstract representation of the system status. The feature design considers two key factor, *i.e.*, the performance of an intrusion detection and the computational complexity during the extraction.

To this aim, we create features directly extracted from a bitstream in the network before decoding. Specifically, counting an occurrence of a bit-symbol in the binary bitstream is employed for forming a high dimensional feature. For example, the occurrences of "1" in the bit-field named "DATA" in a CAN packet syntax is shown in Fig. 3. As the "DATA" contains 8 Bytes (=64 bits), the bits ranging from the 0-th position to the 63-th position are treated as a 64-dimensional feature. The counts of the bits are normalized into a probability. We may use all the bit patterns as a high dimensional feature. However, if necessary, the dimension can be reduced by a prior knowledge regarding a specific semantics in the corresponding syntax element. In the proposed method, the semantics is divided into mode information and value information. The mode information is defined as a state of a vehicle, e.g. adjusting a wheel, characterizing a CAN packet. The value information is constructed with bit-fields determining the value of the specific mode e.g. a wheel angle or a speed. There can be the other bits beside the mode information and the value information, and they will be not included into the training process.

Note the proposed method requires only little computational complexity during the feature extraction because it requires few shift and logical operators handing bits. Any features form the reconstructed original digital signal may yield a better performance, but the complexity is intractable to the real-time detection in the IDS.

C. Training

DNN provides an efficient mean to model nonlinear relations between inputs and outputs while other supervised machine learning algorithms are fitted into linear models.

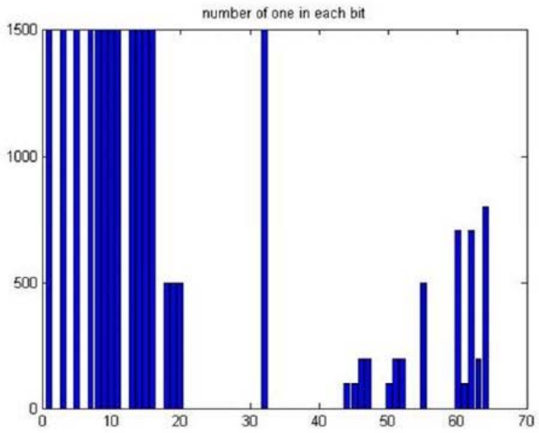


Fig. 3. Proposed CAN packet feature containing the number of "1"-bit.

Hence, we adopt DNN to the intrusion detection problem as our feature from CAN bitstreams have nonlinearity. The DNN can be augmented from the conventional multi-layered artificial neural network (ANN). However, the straightforward augmentation is inefficient when using the back-propagation learning, caused by the vanishing gradient problem and the slow convergent speed. To prevent the problem Erhan *et al.* propose the unsupervised pre-training scheme where the weight parameters in the hidden layers are trained using a Restricted Boltzmann Machine [14]. Motivated by this, we apply the pre-trained parameters as an initial value in the proposed technique.

We show the learning mechanism of the proposed DNN applied to classifying the normal and hacking packets as follows. Fig. 4 shows an input layer, multiple hidden layers, and an output layer. Each node in Fig. 4 calculates an output with using an activation function of an input value. Though advanced activation functions such as rectified linear unit (ReLU) [12] are developed in recent, we use a sigmoid activation for an ease of implementation.

Considering a supervised learning, we have a training set $\{(\mathbf{b}^{(1)},y^{(1)}),\ldots,(\mathbf{b}^{(K)},y^{(K)})\}$ of K samples where $\mathbf{b} = \{b_0,b_1,\ldots,b_B\} \in R^B$ is the set of the feature including B bits, and y is the binary result determining a normal packet or a hacking packet. For this, a cost function C as an mean squared error function between the prediction value and the output is defined as,

$$C(W;\mathbf{b},y)=\frac{1}{2}||h_W(\mathbf{b})-y||^2, \qquad (1)$$

where W is the set of an adaptation weight in an edge, connecting two nodes in the network, and $h_W(\mathbf{b})$ is a hypothesis providing an estimated output. For a batch training, we define the overall cost function to be

$$C(W)=\frac{1}{K}\sum_{k=1}^{K}C(W,\mathbf{b}^k,y^k)+\frac{\gamma}{2}\sum_{n=1}^{N}\sum_{i=1}^{M_l}\sum_{j=1}^{M_{l+1}}(w_{ji}^n)^2, \qquad (2)$$

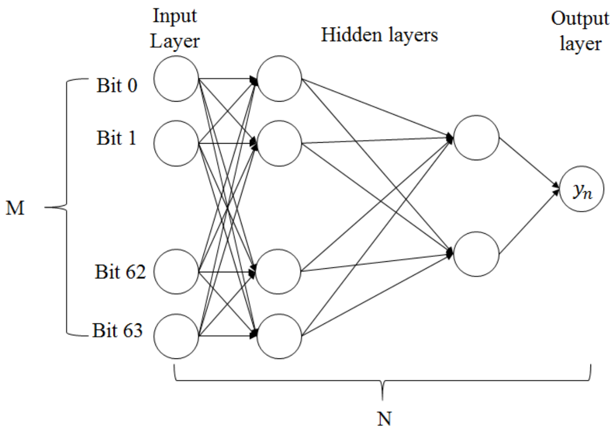


Fig. 4. Architecture of deep neural network.

where N is the depth of the neural network, M_l is the number of nodes in the l-th layer, and $w_{ji}^n \in W$ is the weight of the connection between a node i in the $n-1$ th layer and a node j in the n th layer. In each node in a layer, the output is computed with the sigmoid function of the linear combination of input values and the weights.

We aim to minimize the cost function in (2) to obtain the weighting parameters. The back propagation algorithm is known to be effective with a pre-trained parameters in the problem and, thus, we use a stochastic gradient method to train the network. Specifically, we obtain the partial derivative of the cost function $C_k(W)$ and use the term for the adaptation in each iteration,

$$w_{ji}^{(n)} = w_{ji}^{(n)} + \eta \frac{\partial C_k(W)}{\partial w_{ji}^{(n)}} \tag{3}$$

where η is an adaptation parameter.

D. Detection

In a detection step, we predict a class of a testing CAN packet with the trained deep neural network. The output is simply calculated with the trained wight parameters in W and the extracted feature set $\mathbf{b}$ from the testing CAN packet. The optimum weight parameter set W after training the DNN is stored in a profiling module because the learning requires some computational burden. However, the decision is very quick as compared to the learning process since it consists of a few multiplications to a forwarding direction. Once the input feature is given to the network, the output in the final node is determined as either 0 or 1, telling if the packet is normal or abnormal.

III. Experimental Results

A. Experiment setup

We use a CAN packet generator named Open Car Testbed and Network Experiments (OCTANE) [15] to generate the training and testing packets. The number of the generated packets used for our experiments is about 60,000. We assign 70% packets to the training data and 30% packets to the validation data to avoid any overfitting problem in the training. The number of testing packets is about 3900, which have been not used for the training packets. The number of the normal packets and the attack packets are, respectively, 3450 and 450 among the testing packets. The experiments are conducted in a PC with a 3.4GHz intel CPU.

Identifier		Data Field
400(hex)	• • •	03 yy 00 01 aa bb cc dd (hex)

Fig. 5. Packet frame and syntax element used in the test.

B. Attack Scenario

As in [4], [5] several attack scenarios are assumed in our tests. We assume an attacker to target an instrumental panel to deceive a driver by showing a wrong value of the tire pressure monitoring system (TPMS) on the panel. We determine a syntax of a CAN packet such as an identifier, a control field, a data field, and so on, as shown in [4] but also add some Gaussian noises if the corresponding fields present some analog values. The syntax is shown in Fig. 5 that the identifier is set to a hex number 400, and the data field consists of 8 bytes including some values and modes. In the string of the hex numbers, the second byte position in the "Data Field" represents a TPMS mode (e.g. A0 and 0F, respectively, for the TPMS lamp on and off), and the 5th, 6th, 7th, and 8th bytes positions represent the values of the pressure of four tires, ranging from 0∼45 psi. The other bytes positions are not considered as the inputs to the deep neural model.

In our simulation, the packets are generated to show almost consistent tire pressure with a small random generated noise while the attacker keep injecting an attack packet in the middle of the time to confuse a system. The attack packet aim to modify a TPMS mode and a value of tire pressure. For example, TPMS lamp blinks in the panel even though all the values regarding the tire pressure are actually in safe ranges.

C. Performance Evaluations

We use measurements of a false negative and a false positive to evaluate the classification performance. R_A an R_N are the detection of an attacking packet and a normal packet, respectively, shown in (4) and (5). In the equations, T_A and T_N are, respectively, the total number of the attack packets and the normal packets, and D_A and D_N are, respectively, the number of the detected attack packets and normal packets.

$$R_A(\%) = \frac{D_A}{T_A} \times 100 \tag{4}$$

$$R_N(\%) = \frac{D_N}{T_N} \times 100 \tag{5}$$

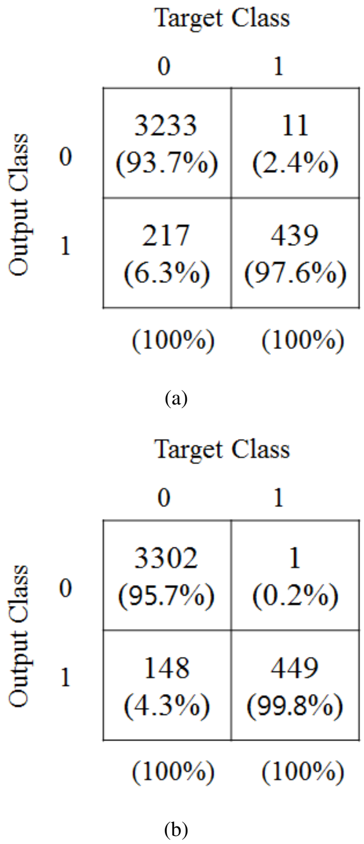
A confusion matrix is presented in Fig. 6 to evaluate the performance of the proposed method. Fig. 6 (a) and Fig.

6 (b) are the confusion matrices when the numbers of the layers are 3 and 5, respectively. The number of the hidden layers represents the layers used for the proposed DNN model. “Target class 0” and “Target class 1” represent the packets belonging to the normal packets and the attack packets, respectively. “Output class 0” and “Output class 1” represent the classification results by the proposed method. As observed, the performance of the proposed method achieves a significantly high detection ratio. In Fig. 6 (a), the false positive error is about 6.3%, and the false negative error is only about 2.4% when the number of the layers is 3. A small false negative error is desired in a safety problem. The false negative error is decreased to 0.2% as shown in Fig. 6 (b) when the number of the layers is 5. The detection performance is consistently high and increases with the number of the layers.

We provide intrusion detection performances and the time complexity in the detection, depending on a different number of hidden layers in Table I. The “Training Time” represents the measurement time required for training the parameters of a DNN structure in a profiling module, and the “Testing Time” represents the measurement time for examining each packet over in-vehicle network. As shown, the detection performance increases with the number of the layers from 95.7% to 99.9%. The time complexity in a training is about a few seconds, and, thus the training should be done off-line. However, the time complexity in a testing time is the only 7∼8 millie seconds for processing 3900 packets, which can be applied to a real-time application.

TABLE I
DETECTION PERFORMANCE AND TIME COMPLEXITY IN A DIFFERENT NUMBER OF LAYERS .

Hidden Layer	R_A %	R_N %	$(R_A + R_N)/2$ %	Training Time(s)	Testing Time(ms/total)
3	97.6	93.7	95.7	4.741	7.957
5	99.8	95.7	97.8	8.469	7.801
7	99.8	99.9	99.9	11.977	8.120



Target Class

Output Class	0	1
0	3233 (93.7%)	11 (2.4%)
1	217 (6.3%)	439 (97.6%)
	(100%)	(100%)

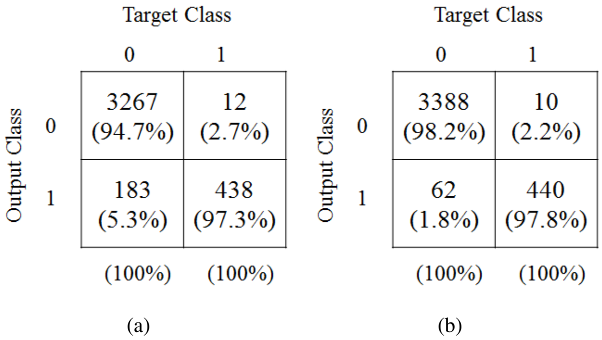
(a)

Target Class

Output Class	0	1
0	3302 (95.7%)	1 (0.2%)
1	148 (4.3%)	449 (99.8%)
	(100%)	(100%)

(b)

Fig. 6. Confusion Matrix Results in the hacking scenario when the number of layers is (a) 3 and (b) 5.



Target Class

Output Class	0	1
0	3267 (94.7%)	12 (2.7%)
1	183 (5.3%)	438 (97.3%)
	(100%)	(100%)

(a)

Target Class

Output Class	0	1
0	3388 (98.2%)	10 (2.2%)
1	62 (1.8%)	440 (97.8%)
	(100%)	(100%)

(b)

Fig. 7. Confusion Matrix Results in the hacking scenario when the number of layers is (a) 3 and (b) 5. As compared with Fig. 6, all byte positions are used for training the DNN.

As aforementioned, the proposed method regards byte positions in the “Data Field”. In this experiment, we evaluate a case when all the byte positions are used in training the deep neural network. A confusion matrix in Fig.7 shows false negative errors about 2.7% and 2.2%, respectively, in cases when the numbers of the layers are 3 and 5. The overall detection performance is similar to that of Fig. 6. However, the proposed method provides a lower false negative error. Furthermore, the proposed method has a lower computational complexity when the input bytes are selectively used for the neural network. In Fig.4, each node corresponds to a byte position in a CAN packet. In Table II, M and m denote the total number of the byte positions and the number of the selective byte positions in the packet, respectively, and L is the number of layers. The complexity of the proposed method in terms of the number of operators and memory requirement is analyzed with a big-O notation. As shown in Table II, the amount of adders, multiplications, and memory are proportional to M^2N and m^2N, depending on the methods. Hence, the proposed method can reduce the computational complexity with a factor of $(\frac{m}{M})^2$.

IV. CONCLUSION

In this paper, we proposed a novel intrusion detection approach using a deep neural network (DNN), training in-vehicle network packets exchanged between electronic control units (ECU). High-dimensional in-vehicle network packets to extract features were trained and used for discriminating normal and hacking packets. The features were directly taken

TABLE II
THE NUMBER OF OPERATORS FOR PROPOSED METHOD AND BLIND METHOD.

Hardware	Using All Byte Fields	Using Selective Byte Fields
Add	$O(M^2N)$	$O(m^2N)$
Multiplication	$O(M^2N)$	$O(m^2N)$
Memory	$O(M^2N)$	$O(m^2N)$

from a bitstream over the network for being high efficiency and low complexity features. Once the features were trained and stored in the profiling module, the proposed system examined an exchanging packet in the vehicular network to decide whether a system was being attacked, or not. The system provided a real-time response to the attack with a significantly high detection ratio from our experimental results.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2056587).

REFERENCES

[1] F. Yu, D.-F. Li, and D. Crolla, "Integrated vehicle dynamics controlstate-of-the art review," in *Vehicle Power and Propulsion Conference, 2008. VPPC'08. IEEE*. IEEE, 2008, pp. 1–6.

[2] K. Pazul, "Controller area network (can) basics," *Microchip Technology Inc*, p. 1, 1999.

[3] S. Biswas, , R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, "Experimental security analysis of a modern automobile," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 447–462, 2010.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Experimental analyses of automotive attack surfaces," in *Proceedings of USENIX Security*, 2011.

[6] C. Miller and C. Valasek, "Adventures in automotive networks and control units," 2013.

[7] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networkspractical examples and selected short-term countermeasures," *Reliability Engineering and System Safety*, vol. 96, no. 1, pp. 11–25, 2011.

[8] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2008.

[9] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," *2010 6th International Conference on Information Assurance and Security, IAS 2010*, 2010.

[10] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1110–1115, 2011.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[12] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 74–82, 2012.

[13] V. Verendel, D. K. Nilsson, U. E. Larson, and E. Jonsson, "An approach to using honeypots in in-vehicle networks," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*. IEEE, 2008, pp. 1–5.

[14] E. D, B. Y, C. A, M. P, V. P, and B. S., "Why does unsupervised pre-training help deep learning," *The Journal of Machine Learning Research*, 2010.

[15] C. E. P. Najafi Borazjani and D. McCoy, "OCTANE: An Extensible Open Source Car Security Testbed," *Proceedings of the Embedded Security in Cars Conference*, 2014.