CANTransfer – Transfer Learning based Intrusion Detection on a Controller Area Network using Convolutional LSTM Network

Shahroz Tariq*
Dept. of Computer Science and Engineering
Sungkyunkwan University
Suwon, South Korea
shahroz@g.skku.edu

Sangyup Lee*
Dept. of Computer Science and Engineering
Sungkyunkwan University
Suwon, South Korea
sangyup.lee@g.skku.edu

Simon S. Woo†
Dept. of Computer Science and Engineeing
Dept. of Applied Data Science
Sungkyunkwan University
Suwon, South Korea
swoo@g.skku.edu

ABSTRACT

In-vehicle communications, due to simplicity and reliability, a Controller Area Network (CAN) bus is widely used as the de facto standard to provide serial communications between Electronic Control Units (ECUs). However, prior research exhibits several network-level attacks can be easily performed and exploited in the CAN bus. Additionally, new types of intrusion attacks are discovered very frequently. However, unless we have a large amount of data about an intrusion, developing an efficient deep neural network-based detection mechanism is not easy. To address this challenge, we propose *CANTransfer*, an intrusion detection method using Transfer Learning for CAN bus, where a Convolutional LSTM based model is trained using known intrusion to detect new attacks. By applying one-shot learning, the model can be adaptable to detect new intrusions with a limited amount of new datasets. We performed extensive experimentation and achieved a performance gain of 26.60% over the best baseline model for detecting new intrusions.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Controller Area Network, Intrusion Detection, Transfer Learning, Convolutional LSTM, In-Vehicle Network

ACM Reference Format:
Shahroz Tariq, Sangyup Lee, and Simon S. Woo. 2020. *CANTransfer* – Transfer Learning based Intrusion Detection on a Controller Area Network using Convolutional LSTM Network. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3341105.3373868

*Equal Contribution
†Corresponding Author

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373868

1 INTRODUCTION

As demonstrated by the Consumer Electronics Show (CES 2018) [5], self-driving cars and related vehicle technologies are one of the fastest-growing areas of interest to many companies. At CES 2018, a total of 556 companies were active in the automotive and vehicle technology sector, including not only major automotive companies such as Ford, Mercedes-Benz, Toyota, and Hyundai, but also other Internet companies such as Amazon, Intel, NVIDIA and Cisco [16]. This area has become an explicit meeting point and a point of a partnership between traditional car manufacturers and the most prominent Internet companies [16]. Also, with considerable interests and developments in the Internet of Things (IoT), we can easily envisage this automotive technology being seamlessly connected to many other IoT devices, clouds, and other cyber-physical systems (CPS) via wireless and cellular networks. Nevertheless, when vehicles link to other networks, there may be periodic network attacks such as Denial-of-Service (DoS), packet injection, and spoofing attacks to target vehicle subsystems. Security researchers have discovered that cyberattacks are used to manipulate data inside the vehicle's internal control bus [15, 28]. This vulnerability could allow the intruder to influence physical subsystems such as the steering wheel, engine, and brakes that could potentially endanger passengers or pedestrians. Additionally, new attack approaches will emerge that exploit particular protocol weaknesses in automotive technology. It is, therefore, necessary to plant and strengthen various parametric defense mechanisms throughout the vehicle to extenuate and mitigate cyberattacks.

The Controller Area Network (CAN) bus [32], which is a de-facto standard for serial communication, has been widely used for in-vehicle communication to provide an effective, stable and economic link between electronic control units (ECUs). The CAN bus is a broadcast-based networking protocol that allows a peak baud rate of up to 1 Mbps on a single bus built by Bosch in 1985. Due to its low prices, size, and functionality, most car manufacturers have embraced the CAN bus. However, due to its simplicity, there are several vulnerabilities in the CAN bus. As demonstrated by Lee et al. [15] and Song et al. [28], the receiving node does not verify the source of a CAN message; hence, many network traffic injection attacks are possible. As a result, numerous network attacks can be easily carried out and practically deployed on the CAN bus. Types of such attacks involve flooding the bus with messages meant to circumvent legitimate ones or using spoofed bus identifiers with

invalid information. Additionally, there are far more sophisticated and stealthy attacks. These attacks seem to be legitimate-looking traffic sequences, and it is hard to distinguish them from regular traffic.

To address these challenges, we need to develop potent detectors and defense mechanisms and evaluate them against various types of intrusions. We propose *CANTransfer*, an intrusion detection method on Controller Area Network using Transfer Learning [27] based on Convolutional LSTM model [35]. The intuition is first to train the Convolution LSTM (ConvLSTM) based model with normal data as well as previously known intrusion dataset as a two-class classification problem. After that, we use one-shot Transfer Learning [10] to re-train the model to detect new intrusion attacks. The reason for explicitly using Convolutional LSTM is that they have previously shown to capture better spatial-temporal details of the data as compared to LSTM based models, achieving high performance [30, 35, 36].

We evaluate our approach with CAN datasets collected from two different cars, KIA Soul and Hyundai Sonata, after driving more than 24 hours with ten different drivers [15, 29].

Our contributions are summarized below:

- **Novel attack detection architecture:** We developed *CANTransfer*, a novel intrusion detection architecture based on ConvLSTM based model, which can detect attacks in multivariate time series data. *CANTransfer* can detect not only known attacks in a CAN bus but also is capable to detect unseen attacks via transfer learning.
- **Evaluation with real datasets**: We evaluated the accuracy of our approach with the CAN dataset collected from two real vehicles, KIA Soul [34] and Hyundai Sonata [33]. We also compare our model with four other baseline models, and demonstrate that we achieve higher detection performance for both known (95.25% vs. 83.3%) and new attacks (88.47% vs. 58.78%) from the best baseline method.
- **Effectiveness of one-shot learning:** In particular, we demonstrate a high accuracy using one-shot learning in detecting different variant of attacks, where our model only requires one sample amount of new intrusion samples to detect a new type of intrusion through various experiments.

Our research shows highly promising results in detecting diverse types of severe vehicle attacks, and it can be integrated into a vehicle as an effective defense mechanism. Also, our approach will help reduce needs for collecting a massive amount of data for detecting each new type of network intrusion.

The remainder of this paper is organized as follows: In Section 2, we discuss the background and recent automotive security research directly relevant to our work. Details of our detection approach and algorithm are explained in Section 3. Then, we show our experiments and results in Section 4. We offer discussion and describe limitations in Section 5 and conclude in Section 6.

2 BACKGROUND AND RELATED WORK

Controller Area Network Bus. The version 2.0 of the Controller Area Network (CAN) specification is a broadcast-based communication protocol [2]. With embedded Electronic Control Units (ECUs) in modern cars, it is the primary means of communication.ECUs handle all aspects of car behavior, while cars typically have two CAN buses: the first type is a high-speed one (about 500Kbps), and the second type is a slower one (about 125Kbps) dedicated to passenger compartment functionality. Data is transmitted on the CAN bus in frames containing an arbitration ID, data payload (0-64 bits), and extra low-level bus control fields, as shown in Fig. 1. There are four types of the standard CAN frame in this protocol. **(1) data frame** for data transmission, **(2) remote frame** for asking a data frame for a destination node, **(3) error frame** for notifying an error in the currently transmitted frame, and **(4) overload frame** for delaying the next message until the current message is processed. The ID and the payload of data in a frame are used as main components to generate attacks in most circumstances. The ID is used for both identification and bus arbitration, where lower IDs are given higher priority when two ECUs simultaneously attempt to transmit. For example, '`0000`' is going to have top priority among all. The data frames, typically in a proprietary format, contain control commands and status information about the automobile state.

Intrusion Detection on CAN bus. Cho and Shin [6] suggested a recursive least square algorithm to spot abnormalities within the in-vehicle network. Boudguiga et al. [3] explored the CAN protocol's lack of security mechanisms, making it susceptible to DoS, impersonation, and isolation attacks. They also suggested a simple method for CAN intrusion detection by making every microcontroller monitor in the CAN bus to detect malicious frames. Muter and Asaj [24] introduced the concept of entropy in the CAN bus and used it to detect anomalies through associating entropy with the reference set. The method is also more systematic and involves sensors to monitor the traffic status for the detection of anomalies. Our work is different from the above study, as we provide the algorithm for intrusion detection, which employs more fine-grained CAN frame features with better generalizability using transfer learning.

Choi et al. [7] have suggested VoltageIDS that can secure CAN network in the vehicle. They mainly use the unique features of an electrical CAN signal in their work as an ECU fingerprint. Hoppe et al. [11, 13] and Miller and Valasek [22] address the detection of intrusion in the vehicle using the number of messages sent over a CAN bus interval. Hoppe et al. [11, 12] introduced simple attacks from the black box attack on an ECU gateway, enabling an attacker to sniff arbitrary internal communications and suggest an IT-Forensic intrusion detection system. Miller and Valasek [21, 23] demonstrated they showed that they were able to take complete control over the automobile from a remote site by using the wireless network, provided the IP address of the vehicle is known. Also the degree of control needed for injecting messages onto the CAN bus and also suggested countermeasures.

Song et al. [28] found that the interval of time between messages is changed during the attack, and they can detect anomalies using this information. Recently, Lee et al. [15] have shown that they can detect intrusions by using the offset response ratio and time interval of remote frames. We compared and demonstrated higher accuracy of our method compared to OTIDS [15]. Recently, Taylor et al.[31] have introduced an attack framework for automobiles that describes the characteristics of cyberattacks in the automotive. We analyzed attacks like a replay attack and found that more extended events can be identified more effectively than shorter ones. This paper combines Taylor and others' work for the replay attack

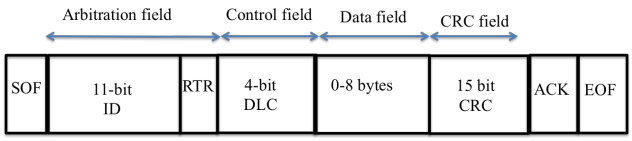


Figure 1: CAN Bus Frame Format

generation. [31]. This paper incorporates the work by Taylor et al. [31] for the replay attack generation.

Convolutional LSTM. Recently, considerable research studies have shown that deep learning for intrusion and anomaly detection on time series data is effective [29, 30, 35, 36]. Detecting network intrusions in the CAN bus by utilizing both the rule-based and RNN approach showed the effectiveness of RNN in this domain [29]. Ensembling, the result from the rule-based method and RNN output result, was able to complement each other and detect different attacks that occurred on the bus with high accuracy. For general-purpose sequence modeling, LSTM is used frequently as a special RNN structure. As an extension of LSTM, Shi et al. [35] introduced Convolutional LSTM (ConvLSTM) for nowcasting precipitation and other forecasting problems. ConvLSTM outperformed fully connected LSTM by having new convolutional structures in both input-to-state and state-to-state transitions. Another work by Zhang et al. [36] used ConvLSTM to detect anomalies in multivariate time series data. They proposed Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) which applies the attention-based ConvLSTM network to capture the temporal patterns. Even for practical operating applications such as Spacecraft multivariate sensor system anomaly detection, ConvLSTM has proved its efficiency and high performance [30]. In this paper, we utilize ConvLSTM to detect in-vehicle communication intrusion attacks by performing prediction on the multivariate stream.

Transfer Learning. Training a neural network for classification tasks is usually done by training with the source dataset from scratch. Also, in many machine learning-based classification models, they assume that the training and testing data must be in the same domain space [26]. Transfer Learning enables the model to transfer knowledge to another unseen domain by reusing the weights from the source model of related tasks or domains [26]. Pratt [27] proposed Discriminability-Based Transfer (DBT) which utilizes predefined weights from the source and rescales the transferred weight with the target. This work demonstrates that DBT-initialized target networks learn significantly faster as compared to networks initialized randomly. Fei-Fei et al. [18] determined the effectiveness of transfer learning by proposing a framework that learns a representation across different domains in a label efficient way. They utilized a metric learning-based approach to simultaneously optimize with labeled source data and unlabeled or sparsely labeled data in the target domain. Another work from Fei-Fei et al. [10], 'One Shot Learning of Object Categories', coined the term one-shot learning and opened the new transfer learning research. Instead of training a classifier from scratch, this paper demonstrates that knowledge from previously learned categories can be used to train a novel category with one or a handful of images. This paper presented a variation on a Bayesian framework for representation learning for object categorization, which produces informative models when the number of new training examples is extremely small. Cozzolino et al. [9] also used a similar concept of transfer learning to detect fake or forged images from multiple manipulation approaches. When training with data from one specific forgery method, they show that Convolutional Neural Networks detect examples extremely well. Their work uses the auto encoder-structured model, ForensicTransfer, which enforces activations in different parts of a latent vector from the encoder for the real and fake classes. The learned latent vector acts as a form of anomaly detector. ForensicTransfer shows significant improvements in transferability by training from source and transfer learn the same model to a minimal target dataset. Similarly, our *CANTransfer* trains the model with known intrusion, and then transfer learns to new intrusion. However, our approach uses ConvLSTM to detect attacks in time series.

Table 1: Sample CAN packets, showing different ID and DLC fields

Timestamp	ID	DLC	Data
1479246664.162438	0153	8	00 21 10 ff 00 ff 00 00
1479246664.164967	02b0	5	bd ff 00 07 dc
1479246664.165822	043f	8	00 40 60 ff 58 28 08 00
1479246664.171065	05f0	2	f4 00
1479246664.171695	0002	8	00 00 00 00 00 06 01 a2

3 OUR PROPOSED MODEL - *CANTransfer*

We develop the ConvLSTM-based *CANTransfer* to address the following research questions (RQs): **(RQ1) Intrusion detection performance.** Can ConvLSTM outperform over other state-of-the-art intrusion detection methods for CAN bus? **(RQ2) Importance of preprocessing.** Does the preprocessing and data transformation matter for intrusion detection performance? Which preprocessing method works the best for ConvLSTM? **(RQ3) Transfer Learning.** Does one-shot transfer learning increase the detection rate and more importantly, does it help in reducing the false-positives (FP)? **(RQ4) Improvement from zero-shot.** How does one-shot compare against zero-shot? **(RQ5) False-positive.** How does the FP-rate affect the performance? **(RQ6) Processing Time.** How much processing time does *CANTransfer* take to detect the intrusion in the worst-case scenario? **(RQ7) Loss and Accuracy.** Is the loss for the *CANTransfer* model converging with time, and how does the accuracy look like for transfer learning? In order to address these questions, we describe the details of *CANTransfer* in this section.

CAN Traffic Analysis. In order to better understand normal CAN bus traffic during a standard car operation, we first studied high-speed CAN bus traffic. Such data helps us to recognize the regular and frequent CAN bus traffic. In order to understand their different behavior, we use two different cars, KIA Soul (car 1) and Hyundai Sonata (car2). We discovered that each car uses different frame IDs for the same operations; however, this comportment is anticipated as these cars are manufactured by different companies. There is, however, a high correlation between the occurrences and sequences of priority bits between Car 1 and 2. For example, in both vehicles, the sensor IDs used for the vehicle braking system vary, but they have a higher priority on the bus than other sensors.

Table 2: Representation of the dataset after preprocessing.Labels are normal (0) and intrusion (1).

Time Diff.	ID	DLC	Data								*Label*
			D1	D2	D3	D4	D5	D6	D7	D8	
0.6	497	8	0	0.501	0.062	1	0	1	0.815	0.368	**0**
0.4	544	8	0	0	0	0	0	0	0	0	**0**
0.6	848	8	0.874	0.011	0.996	0.011	0.035	0	0.227	0.062	**0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.2	399	8	0.019	0.141	0.368	0.054	0.141	0.101	0	0.498	**1**
0.1	128	8	0	0.109	0.141	0	0	0.356	0	0.062	**1**

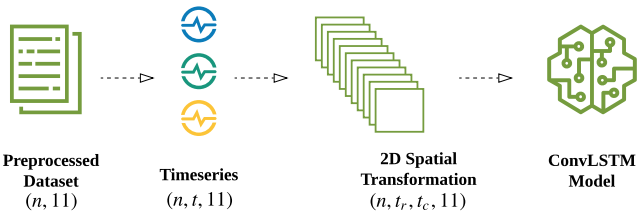


Figure 2: Dataset is transformed to be processed with the Convolutional LSTM model.

Table 1 displays a sample of different CAN-packets where a column of 1st represents packet-arrival time timeline, 2nd column includes sender's ID (ECU), the total amount of bytes occupying a data field of the packet is indicated in the 3rd column and the data field is shown in 4th column. We observed various ID, DLC, and Data values for multiple CAN packets, as shown in Table 1. We found 45 unique sensor IDs for Car 1 and 29 unique frame IDs for Car 2, where each ID regularly transmits data between 10ms and 1s. Although most IDs had different data load values for the full scope of our data collection periods, a small group of data fields used in some IDs was almost unchanged. For example, ID '`02c5`' in Car 2 has a DLC of '8' (i.e., the data field contains 8 bytes) and '`ac 16 00 00 00 71 00 00`' is a data field. We observed that in the whole dataset only the **4**th,**11**th and **12**th byte in the data field of '`02c5`' were changing and every other byte remained the same. The data field '`ac 19 00 00 00 e2 00 00`' is another instance with ID value '`02c5`', which exhibited the same behavior. Additionally, several other IDs show similar patterns of operation on different bytes. These patterns are, therefore, useful for modeling the normal behavior of the CAN packet, then allow us to spot irregularities and target traffic.

We used an algorithm by Markavitz and Wool [20] to obtain a more in-depth insight into the design of the CAN frames for examining information on the data field bits and classifying them further. This algorithm classify neighboring bits as **(1) constant fields** are those that are not changing, **(2) multivalued fields** are those that take a small number of relative values, and **(3) sensor fields** are those that take many different values. We also added a strict counter detector to the algorithm for IDs with non-recurring growing values by using the approach of Taylor et al. [31]. We categorized the sensor field by its variance, and we observed three clusters based on the number of unique field values that we named low (under 65), medium (between 115 and 475), and high (over 1800). For example, the IDs `0044`, `051a` and `0018` are low, `0587`, `04f0` and `01f1` are medium, and `0153`, `0220` and `0164` are high. These results are comparatively consistent with Taylor et al. [31] observations. After examining the normal CAN bus traffic sequences, we summarize our two main findings as follows:

- The majority of the identifiers produce seemingly random data without any clear general representation, whereas there are some comparisons between the relative ordering of priorities in the identifiers. Moreover, the consistent creation of new data fields implies that data sequences can not be depicted as a univariate sequence of symbols, limiting the usefulness of most sequence-based anomaly detection methods.
- The static rate of the CAN bus frames in both cars allows detecting abnormalities and attacks at arrival rates using frequency-based algorithms.

Preprocessing. The dataset in its natural state is not adequate for training a neural network-based method, so we have to perform some preprocessing. We split and transformed the original 4-feature dataset from Table 1 into an 11-features dataset, as shown in Table 2. The 'Time Diff.' column is the difference between two consecutive timestamps formally known as an interpacket delay. The Data column is subdivided into eight columns (D1-D8), where each column contains two hexadecimal values. Also, the hexadecimal values in ID and Data columns (D1-D8) are converted to decimal values.

Time series Transformation. We transformed the original dataset and constructed a multivariate time series. A multivariate time series data means we have multiple observations for each timestep. Now, we can develop a classifier, where input is time series and an output label dependent on the input time series (normal vs. attack). A LSTM model needs sufficient context to learn a mapping from an input sequence to an output value, and LSTM can support parallel input time series as separate variables or features. Therefore, we need to split the data into samples, maintaining the order of observations across the 11 features' input sequences. If we choose t input timesteps, then the sample will have this $(t, 11)$ shape. If there are n samples generated from the dataset, then it will have a time series with the following $(n, t, 11)$ shape as shown in Fig. 2.

2D Spatial Transformation. Next, we transform our time series into a 2D space as shown in Fig. 2 to be preprocessed with a Convolutional LSTM. Hence, the timesteps t are divided into a two-dimensional multivariate time series where t_r and t_c represent the rows and column of this new space, as shown in Fig. 2. We applied a constraint on t to be even, so that after splitting t, we will always have equal size for t_r and t_c. Now, the data is ready to be processed by our Convolutional LSTM model.

Convolutional LSTM. The core component of our intrusion detection approach uses a multivariate Convolutional LSTM (ConvLSTM) [35]. By learning from the normal and abnormal data, the goal of our *CANTransfer* model is to predict between normal and abnormal sequences effectively via transfer learning. In our approach, a single model is created for an entire system. As observed by Hundman et al. [14], LSTMs struggles to accurately predict n-dimensional outputs with a large n, preventing the input of a multivariate stream into one or a few models. However, Convolutional LSTM (ConvLSTM) has shown to overcome these limitations, and thus it is suitable for predicting multivariate time series data as shown by [30, 36]. In order to avoid overfitting, an early-stopping mechanism is applied when validation error starts increasing [4].

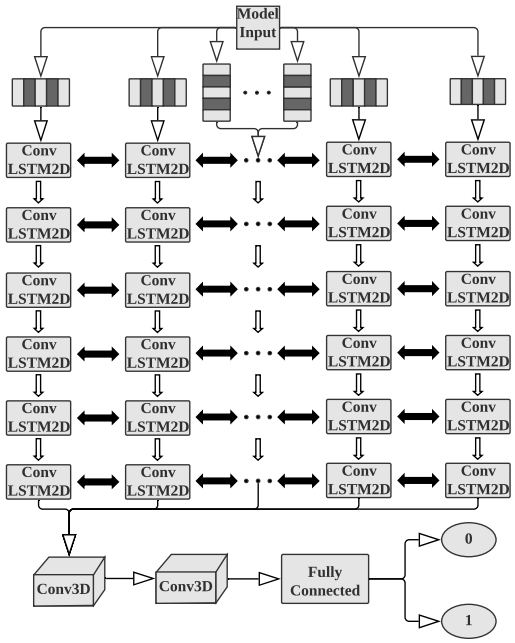


Figure 3: A graphical representation of our *CANTransfer* model.

A visual representation of our *CANTransfer* model is provided in Fig. 3.

Multivariate prediction. The dataset is transformed into a two-dimensional spatial-temporal data X containing t historical values of each feature f. The length of the sequence t determines how far we look into the past for future predictions. The time series for some feature f_i is given in Eq. (1):

$$T(f_i) = \{f_i(t_1), f_i(t_2), f_i(t_3), \ldots, f_i(t_n)\}, \tag{1}$$

where any $f(t_j)$ is the jth step in the time series $T(f_i)$. Hence, the timeseries for all 11 features can be written as follows:

$$T\left(f_{1,\ldots,n}\right) = \{T(f_1), T(f_2).T(f_3), \ldots, T(f_n)\} \tag{2}$$

By expanding Eq. (2) we obtain X, which is the input for *CANTransfer* model as shown in Fig. 4

Transfer Learning using *CANTransfer*. Deep learning methods are data-hungry by nature. Hence, they need massive amount of training examples to learn the weights. This data-hungry nature is one of the limiting aspects of deep neural networks. One-shot learning is a variant of transfer learning, where we aim to infer the required output based on just one or a few training examples. This characteristic is mainly helpful in real-world scenarios, where it is not easy to have labeled data for every intrusion attack as well as newly added intrusion classes, in addition to labels normal data. Once the *CANTransfer* model is fully trained, we use one-shot Transfer Learning [10]. The advantage of using one-shot learning is that we need to train only one example of a particular intrusion to successfully detect all instances of it.

$$X = \left\{ \begin{bmatrix} f_1(t_{r_1,c_1}) & \ldots & f_1(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_1(t_{r_n,c_1}) & \ldots & f_2(t_{r_n,c_n}) \end{bmatrix} \begin{bmatrix} f_2(t_{r_1,c_1}) & \ldots & f_2(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_2(t_{r_n,c_1}) & \ldots & f_2(t_{r_n,c_n}) \end{bmatrix} \vdots \begin{bmatrix} f_{11}(t_{r_1,c_1}) & \ldots & f_{11}(t_{r_1,c_n}) \\ \vdots & \vdots & \vdots \\ f_{11}(t_{r_n,c_1}) & \ldots & f_{11}(t_{r_n,c_n}) \end{bmatrix} \right\}, \; y = \left\{ \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right\}$$

Figure 4: The input and output representation for *CANTransfer* model.

4 EXPERIMENTAL RESULTS

We conduct extensive experiments to answer the following seven research questions (RQ1–RQ7) described earlier.

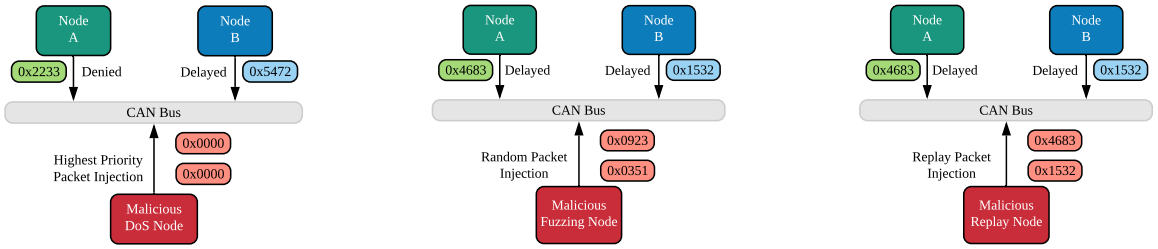
4.1 Experimental Setup

CAN bus data. In order to better utilize the raw dataset of Car 1 and Car 2, we performed preprocessing and normalization on the dataset to produce 11 features. The timestamp is converted into time difference, which indicates the time interval between the current and previous packet. Next, we obtained eight subfeatures by dividing the data payload. For normalization, we divided payloads hex values with '`ff`' which is the maximum value of the raw payload data. Labeling is performed by using timestep t. For example, if t is 16 packets and that window t is in "Normal state duration", all those windows' packets are labeled as 0 for normal traffic and 1 for intrusions. Table 2 shows the preprocessed and normalized dataset. Each packet has 11 features, and the last column is the label. The majority label in window t is given as the label of that t. We divided the collected dataset from each car to 70% for training (2,887,500 packets for Car 1 and 2,625,554 packets for Car 2), 25% for validation (1,031,250 packets for Car 1 and 937,698 packets for Car 2) and 5% for the testing of attack and anomaly detection model (206,250 packets for Car 1 and 187,539 packets for Car 2). Due to the comparatively slow data rate of the CAN bus, all the detectors can run in real-time speed; nevertheless, this might require the installation of dedicated hardware in a car. We tried different values between $2 - 100$ for input timestep t, where t is even.

Machine Configurations. The PC specifications include Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz, NVIDIA GeForce GTX 1060 GPU and Linux 16.04 for the operating system. For developing the algorithm, we used Python 3.6 with the Keras library for deep learning methods by Chollet et al. [8] and TensorFlow [1].

Baseline methods. We compared our *CANTransfer* method with four baseline methods as follow:

- **Threshold-based method:** We conducted a direct comparison with the state-of-the-art OTIDS intrusion detection algorithm by Lee et al. [15], where OTIDS uses the offset ratio and time interval between request and response messages in CAN message to detect attacks.



(a) DoS Attack: Spams the network with a large amount of high priority packets IDs, which results in denying or delaying access to other nodes.

(b) Fuzzing Attack: Injects a large amount of packet into the network using randomly generated packet IDs and delays access to other nodes.

(c) Replay Attack: Injects some previously sniffed traffic patterns into the network, making it look like regular traffic and delays the legitimate traffic.

Figure 5: Network-based attacks in CAN bus.

Table 3: Performance comparison of *CANTransfer* with the baseline methods. The best performer from baseline methods for each metrics is underlined and overall best performers are shown in bold.

Method	Precision (%)		Recall (%)		F1-Score (%)	
	Known	*New*	*Known*	*New*	*Known*	*New*
OCSVM	35.43	10.83	71.15	35.12	47.30	16.55
IF	43.58	15.62	73.42	31.56	54.69	20.90
OTIDS	**99.82**	70.81	71.68	42.01	83.44	52.73
RNN+Heuristics	98.69	70.25	**99.49**	50.53	**99.09**	58.78
CANTransfer	94.93	**87.97**	95.57	**88.97**	95.25	**88.47**
Gain	-4.89	**17.16**	-3.92	**38.44**	-3.84	**26.69**

Table 4: Test result comparison of known vs. new intrusion with and without using Transfer Learning in *CANTransfer*, where class 0 is normal and class 1 is new attack.

		Known Intrusion			New Intrusion		
Zero-shot Learning	**Class**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
	0	99.0%	99.0%	99.0%	68.0%	100.0%	81.0%
	1	99.0%	99.0%	99.0%	67.0%	0.00%	1.00%
One-shot Learning	**Class**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
	0	99.0%	91.0%	95.0%	92.0%	91.0%	91.0%
	1	90.0%	99.0%	94.0%	81.0%	83.0%	82.0%
Gain	**Class 0**	0.00%	-8.00%	-4.00%	24.0%	-9.00%	10.0%
	Class 1	-9.00%	0.00%	-5.00%	14.0%	83.0%	81.0%

- **Classification-based models:** We used One-Class SVM (OCSVM) [19] as a One-Class classification method and Isolation Forest (IF) [17]. Both of these methods learn the decision boundary from the training set.
- **Ensemble-based model:** We also compared against a Recurrent Neural Network (RNN) with Heuristics based approach (RNN + Heuristics) which is an ensemble of RNN with a threshold based method [29].

We use precision, recall, and F1-score for the evaluation and comparison. The input data representation for each of these baselines methods is different. Hence, we transformed and preprocessed dataset tailored for each baseline method.

***CANTransfer* Training.** First, we trained our Convolutional LSTM based *CANTransfer* model with normal data (class 0) and multiple instances of a known intrusion (class 1). The known intrusion, in our case, is a Denial of Service (DoS) attack. Next, we used our trained *CANTransfer* model and performed transfer learning by training with only one instance of a new intrusion (one-shot learning). The new intrusion can be a Fuzzing attack and a Replay attack, where a visual description of DoS, Fuzzing, and Replay attack is shown in Fig. 5. The reason for choosing the DoS attack as a known intrusion is that it is the most common intrusion and relatively straightforward for the model to learn it because of the high volume of attack traffic. Additionally, we designed our model to learn the underlying fundamental characteristics if the DoS attack, not over-fitting, and then adapted to other types of intrusion through transfer learning. By doing so, we achieved better and robust performance against other types of attacks.

4.2 Results

(RQ1) Intrusion detection performance. The precision, recall, and F1-score of different baseline methods against *CANTransfer* are reported in Table 3, where the highest performance values are highlighted in bold, and the best baseline performance values are indicated by underline. The "Gain" row in Table 3 reports the improvement of *CANTransfer* over the best baseline method. In Table 3, we observe that (a) after one-shot learning on our *CANTransfer* achieves higher precision (87.97%), recall (88.97%) and F1-score (88.47%) for 'new' intrusions as compared to threshold (OTIDS), Classification (IF, OCSVM) and Prediction (RNN+Heuristics) based models. Isolation Forest (IF) and OneClass SVM (OCSVM) show low precision and high recall, indicating that most of the predicted labels are incorrect (false-positives) when compared to the training labels. (b) It is interesting to note that with slightly lower F1-Score (95.25%) than the best baseline method (99.09%) on known intrusions, our *CANTransfer* model became better at generalizing the intrusion behavior after one-shot learning as shown in column 'F1-score (*New*)' in Table 3. (c) *CANTransfer* performs the best among all the evaluated methods, yielding an overall F1-score of 95.25% and 88.47% for known and new intrusions. The gain of our approach over the best baseline (RNN+Heuristics) measured up to 26.69% in F1-score. (d) A high recall (88.97%) and precision (87.97%) score of

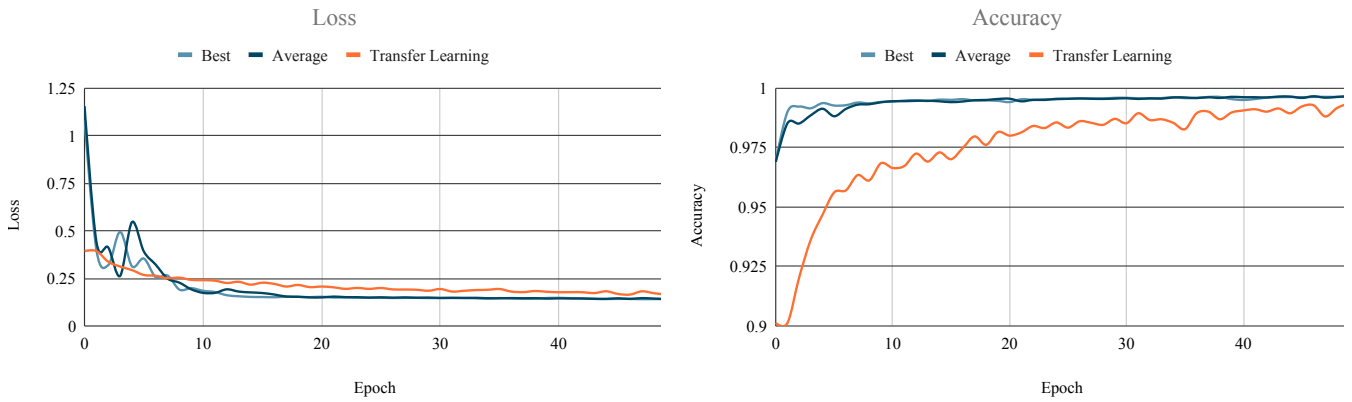


Figure 6: Loss (left) and Accuracy (right) of *CANTransfer* during training.

CANTransfer show that it can detect most of the intrusions with a very few numbers of false-positives, as shown in Table 3.

(RQ2) Importance of preprocessing. In our experiments, we tried two different methods of training our *CANTransfer* method. In the first method, we did not perform any preprocessing or spatial transformation on the raw dataset and allowed to model to train on raw data. In the second method, we preprocessed the dataset and the applied 2D spatial transformation, as described in Section 3. The results are very drastically different in favor of the second method by 72.54% in gain on F1-score, which is what we were expecting. Due to space limitation, we do not present the whole result.

(RQ3) Transfer Learning. From Table 3, we can observe that the F1-Score (88.47%) of *CANTransfer* for new intrusions was increased by 26.69% after applying transfer learning as compared to F1-Score (58.78%) of the best baseline (RNN+Heuristics). As shown in Table 4, the false-positive rate for *CANTransfer* decreased as the precision and recall score for intrusions (class 1) increase from 67.0% (precision) and 0.00% (recall) in zero-shot learning to 81.0% (precision) and 83.0% (recall) in one-shot learning. Therefore, *CANTransfer* achieved a gain of 81.0% in F1-score.

(RQ4) Improvement from zero-shot. We also compared the performance of *CANTransfer* with zero-shot learning, where not even a single sample of new intrusion is given to the model during its training. The comparison can be seen in Table 4. The F1-score for class 0 (91.0%) and Class 1 (82.0%) on new intrusion with one-shot learning is far better than the F1-score for Class 0 (81.0%) and class 1 (1.00%) on new intrusion with zero-shot learning.

(RQ5) false-positives. From Table 4, we can observe that the precision for known intrusion is very high for both zero-shot (Class 0: 99.0%, Class 1: 99.0%) and one-shot learning (Class 0: 99.0%, Class 1: 90.0%). However, for new intrusions, there is a significant difference the precision of zero-shot (Class 0: 68.0%, Class 1: 67.0%) and one-shot learning (Class 0: 92.0%, Class 1: 81.0%). This difference represents the decrease in the false-positives rate after one-shot learning is applied, which shows the effectiveness of *CANTransfer*.

(RQ6) Processing Time. Processing time is a considerable concern in real-time systems. Hence, a method that takes a lot of processing time, will never be feasible. During our experiments, we also checked the processing time of each prediction request sent to *CANTransfer*. To keep the processing time to minimal, we wanted a very small timestep (window size) t. We experimented with multiple sizes of t and concluded that t =16 is the right size as any window size larger than 16 has very similar performance on *CANTransfer*. We also experimented with timestep t from 1–50 and observed that all of the prediction requests were quickly processed in real-time.

(RQ7) Loss and Accuracy. In order to prove that the training process is stable and it will always converge, we ran the same experiments for multiple times and reported the average case and best case. From Fig. 6, we can observe that for both the best and average case, the *CANTransfer* loss values are very similar. The same goes for the accuracy values as well. This similarity demonstrates that the model will generally perform in the same manner. We can also observe from Fig.6, that the loss and accuracy of *CANTransfer* converge to the same point even after one-shot learning. This shows that our *CANTransfer* has successfully learned about the new intrusion via one-shot learning, while maintaining the knowledge of previously known intrusions.

5 DISCUSSION AND LIMITATIONS

What types of proactive actions and countermeasures should be performed after detecting different types of attacks? Although we did not consider in the current research, we believe that an integrated safety and defense system can be developed, providing the following additional countermeasures after detecting an intrusion: (1) Automatically put the car to safety mode. (2) Slow down the car and alert the driver. (3) Send it the attack signature to the cloud for further analysis. For example, a service such as General Motors subscription-based communications, in-vehicle security, emergency services, hands-free calling, turn-by-turn navigation, and remote diagnostics systems known as OnStar [25] can be alerted on detecting an intrusion attack. Another critical factor is the effects of delay in detecting the intrusion on properties such as stability and safety of cars. In this work, we considered the processing time (delay in detection) of our *CANTransfer* method. In our experiments, we were able to perform intrusion detection in real-time. The processing time for the analysis of packets is kept to a minimum, which enables our proposed approach to detect intrusions in real-time. However, there can be some implications, when implementing such systems in a real-world environment, and more research is required on the implementation side.

6 CONCLUSION

During the vehicle operation, intrusions can be dangerous threats and have real serious ramifications and harm to a driver, passengers, pedestrians, and a vehicle. New intrusions are discovered every month, and it is hard to train a neural network for new intrusion until we have a significant amount of data. Hence, we proposed *CANTransfer*, the first line of defense against new intrusion attacks using one-shot transfer learning. We demonstrate the effectiveness of *CANTransfer* in detecting the new intrusions. For future research, *CANTransfer* can be implemented, deployed, and tested in actual vehicle systems and detection performance can be compared. In addition, we should consider the actionable and proactive attack mitigation strategy after detection of different intrusions.

7 ACKNOWLEDGEMENT

We thank KISA and KIISC for the release of CAN dataset. This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (No. 2019M3F2A1072217) and the Basic Science Research Program through the NRF of Korea funded by the Ministry of Science, ICT (No. M2017R1C1B5076474). In addition, this work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Robert Bosch et al. 1991. CAN specification version 2.0. *Rober Bousch GmbH, Postfach* 300240 (1991), 72.

[3] Aymen Boudguiga, Witold Klaudel, Antoine Boulanger, and Pascal Chiron. 2016. A simple intrusion detection method for controller area network. In *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 1–7.

[4] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*. 402–408.

[5] CES. 2018. Consumer Electronics Show. Retrieved September 4, 2018 from https://ces.tech/

[6] Kyong-Tak Cho and Kang G Shin. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection.. In *USENIX Security Symposium*. 911–927.

[7] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee. 2018. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security* 13, 8 (Aug 2018), 2114–2129. https://doi.org/10.1109/TIFS.2018.2812149

[8] François Chollet et al. 2017. Keras. https://keras.io.

[9] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. 2018. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510* (2018).

[10] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence* 28, 4 (2006), 594–611.

[11] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2008. Security threats to automotive CAN networks–practical examples and selected short-term countermeasures. *Computer Safety, Reliability, and Security* (2008), 235–248.

[12] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2009. Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 145–158.

[13] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2011. Security threats to automotive CAN networks?Practical examples and selected short-term countermeasures. *Reliability Engineering & System Safety* 96, 1 (2011), 11–25.

[14] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. *arXiv preprint arXiv:1802.04431* (2018).

[15] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. 2017. OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame. In *Privacy, Security and Trust (PST)*.

[16] Ari Levy and Lora Kolodny. 2018. Self-driving cars take over CES: Here's how big tech is playing the market. https://www.cnbc.com/2018/01/12/intel-cisco-and-amazon-introduce-self-driving-car-technology-at-ces.html. [Online; accessed 4-September-2018].

[17] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–422.

[18] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. 2017. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*. 165–177.

[19] Larry M Manevitz and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of machine Learning research* 2, Dec (2001), 139–154.

[20] Moti Markovitz and Avishai Wool. 2017. Field classification, modeling and anomaly detection in unknown CAN bus networks. *Vehicular Communications* 9 (2017), 43–52.

[21] Charlie Miller and Chris Valasek. 2013. Adventures in automotive networks and control units. *DEF CON* 21 (2013), 260–264.

[22] Charlie Miller and Chris Valasek. 2014. A survey of remote automotive attack surfaces. *black hat USA* 2014 (2014).

[23] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015 (2015).

[24] Michael Müter and Naim Asaj. 2011. Entropy-based anomaly detection for in-vehicle networks. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 1110–1115.

[25] OnStar. 2018. OnStar In-Vehicle Safety and Security. https://www.onstar.com/us/en/home/. [Online; accessed 26-December-2018].

[26] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[27] Lorien Y Pratt. 1993. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*. 204–211.

[28] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 2016. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 63–68.

[29] Shahroz Tariq, Sangyup Lee, Huy Kang Kim, and Simon S Woo. 2018. Detecting In-vehicle CAN Message Attacks Using Heuristics and RNNs. In *International Workshop on Information and Operational Technology Security Systems*. Springer, 39–45.

[30] Shahroz Tariq, Sangyup Lee, Youjin Shin, Myeong Shin Lee, Okchul Jung, Daewon Chung, and Simon S Woo. 2019. Detecting Anomalies in Space using Multivariate Convolutional LSTM with Mixtures of Probabilistic PCA. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2123–2133.

[31] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. 2018. Probing the Limits of Anomaly Detectors for Automobiles with a Cyberattack Framework. *IEEE Intelligent Systems* 2 (2018), 54–62.

[32] Wikipedia contributors. 2018. CAN bus — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=CAN_bus&oldid=857951504. [Online; accessed 4-September-2018].

[33] Wikipedia contributors. 2018. Hyundai Sonata — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Hyundai_Sonata&oldid=857139549. [Online; accessed 4-September-2018].

[34] Wikipedia contributors. 2018. Kia Soul — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Kia_Soul&oldid=857653915. [Online; accessed 4-September-2018].

[35] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*. 802–810.

[36] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1409–1416.