NeuroCAN: Contextual Anomaly Detection in Controller Area Networks

Prashanth Balaji and Majid Ghaderi
Department of Computer Science, University of Calgary
{prashanth.balaji, mghaderi}@ucalgary.ca

***Abstract*—The Controller Area Network (CAN) is an established standard for inter-connecting onboard Electronic Control Units (ECUs) in a vehicle. Through sensors and actuators, ECUs maintain critical vehicle functions such as transmission and engine control. However, security was never a part of CAN design and hence ECUs are susceptible to a wide range of attacks. Thus, in recent years, several anomaly detection systems have been proposed for the CAN bus in order to detect anomalies caused by adversarial attacks or misbehaving sensors. These systems generally try to detect deviations from individual sensor's expected behavior. As such, they are ineffective against attacks that target multiple sensors to accomplish a collective desired behavior without changing the expected behavior of each individual sensor. In this paper, we focus on detecting such attacks by identifying contextual CAN anomalies in real-time. To this end, we present NeuroCAN, a deep learning-based detection system that utilizes Linear embeddings and Long Short Term Memory (LSTM) units to learn the spatio-temporal correlations among sensor data on the CAN bus at a frame level. By exploiting such correlations, NeuroCAN is able to detect contextual anomalies that are otherwise difficult to detect by analyzing individual sensor data. We evaluate NeuroCAN using two publicly available CAN datasets and compare it against existing approaches. Our results show that NeuroCAN achieves over 95% detection accuracy and performs significantly better than the existing baselines.**

I. Introduction

Safe and efficient operation of a modern automobile relies on the Electronic Control Units (ECUs) that maintain critical vehicle functions such as transmission and engine control, lane assist and collision warning. These units are inter-connected over a central bus that forms a backbone through which data is exchanged in real-time. The increasing complexity of the software running on these units has pivoted the view of vehicles from being a mere combination of mechanical components to complex embedded systems capable of handling multiple interactions simultaneously. The incorporation of wireless technologies such as WiFi and Bluetooth has also enabled vehicle-to-everything (V2X) communication, as shown in Fig. 1, to support onboard navigation systems, mobile interfaces and more [1]. The increased dependency on these internal systems has been further driven by the addition of intelligent features such as autonomous driving.

However, such integrations have resulted in a wider attack surface for malicious actors to leverage. The inherent design flaws if unnoticed, can be successfully exploited to gain access or worse create a cascade of system failures when the vehicle is in operation. The Controller Area Network (CAN) is widely used for in-vehicle networks due to its low cost and centralized operation [2]. But it lacks basic security measures thereby permitting attackers to easily gain unauthorized access to various sub-systems onboard a vehicle. Specifically, CAN lacks any authentication measures to verify the identity of the sender of messages. Thus, an attacker can inject carefully crafted messages either directly through the On-Board Diagnostics port (OBD) or through other external interfaces such as the infotainment or wireless communication system [3] to manipulate the vehicle behavior. Though message authentication can be enforced by methods discussed in [4] [5], they either involve altering the existing standard or by attaching additional devices. Such approaches not only introduce additional cost and overhead, but also may create compatibility issues with the original CAN specification [6].

Message injection attacks on the CAN bus either focus on flooding the bus with higher priority messages leading to a DoS attack or by injecting certain payloads to cause the desired effect. The effect of such attacks is that more messages are seen on the bus than usual and therefore can be detected by exploiting specific characteristics of the CAN bus. A few such characteristics include: the frequency of IDs on the bus [7], clock skews of ECUs [8], and differences in entropy [9]. Apart from these approaches, the rapid rise of deep learning has paved the way for "anomaly detection" systems to precisely model the behaviour of an underlying system. Prior works employing deep learning for anomaly detection on the CAN bus are found in [10], [11]. However, attackers have become more creative with their methods to elude detection by performing malicious activities while still adhering to the norms of the system. These attacks are more subtle and can escape the purview of any deployed anomaly detection system that either fails to analyze the payload or that does not perform a collective analysis of the context of the system. For example, an adversary who gains access to the engine control unit could boost the engine RPM while the vehicle is idle. By looking at the context of the system, i.e. vehicle is in idle state, this condition can still be flagged as anomalous even if the RPM is within the normal limits.

In this work, we focus on detecting these types of stealthy attacks that can be identified only when considering the collective behavior of the CAN bus. The basis of our approach is guided by the reasoning that in a correlated system where

This work was supported by Wedge Networks Inc., Alberta Innovates and Natural Sciences and Engineering Research Council of Canada.

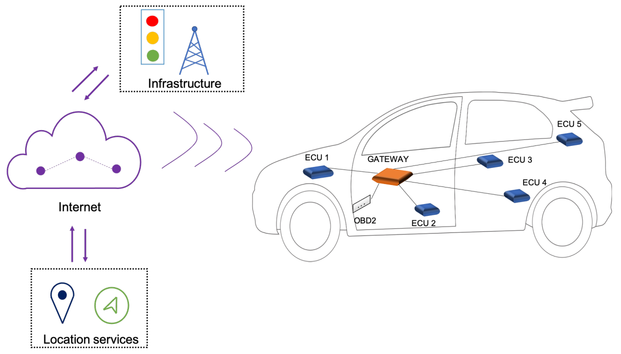


Fig. 1: High-level network topology of CAN.

data is driven by the occurrence of physical phenomena, modifying an aspect of the system even within its limits would cause a deviation to its overall state. While the previously mentioned approaches are capable of detecting DoS, playback and fuzzy attacks [10], [11], they fail to address this particular type of attack that serves as our motivation. To this end, we present NeuroCAN, a deep learning-based detection system that utilizes Linear embeddings and Long Short Term Memory (LSTM) units to learn the spatio-temporal correlations among sensor data on the CAN bus at a frame level. By exploiting such correlations, NeuroCAN is able to detect contextual anomalies that are otherwise difficult to detect by analyzing individual sensor data.

The main contributions of this work are:

1) We present NeuroCAN, a deep learning model that exploits the correlations between sensor readings to identify contextual anomalies in an automotive CAN bus. Our model works with raw CAN data and does not require signal decoding.
2) We train and evaluate the proposed model with frame-level CAN bus traces rather than with decoded signal values as the translation rules are proprietary to every manufacturer.
3) We also train a baseline for comparison and further perform additional experiments to comprehend the impact of the context provided by other ECUs in the detection process. Our results show that NeuroCAN achieves over 95% detection accuracy and performs significantly better than the baseline, which does not consider spatio-temporal sensor correlations.

The rest of the paper is organized as follows. Section II provides an overview of the CAN protocol. We review some of the closely related work in Section III. In Section IV, we present the design of NeuroCAN. We discuss the dataset specifications, evaluation metrics and the obtained results in Section V. Finally, Section VI concludes the paper.

II. CAN Overview

CAN Bus. A modern car comprises 50 to 70 ECUs that are connected through the CAN bus along a central gateway [12]. An ECU can be connected to one or more sensors and actuators. Inputs from the sensors are utilized to drive the actuators to perform the necessary operation through the implemented firmware logic. Typically, two CAN buses are present on-board, one with a higher bandwidth of 500kbps for carrying time-critical data and another with 125kbps for data related to the passenger compartment and other fault-tolerant systems [11]. Unlike most systems where a logic high is driven by 1 and logic low is driven by 0, CAN implements inverted logic states where a 0 on the bus drives high and a 1 drives low. These states are formally termed as dominant and recessive in the CAN standard [6]. The structure of the dataframe implemented in the standard CAN protocol is shown in Fig. 2. As illustrated, a CAN frame comprises ten sections which are as follows: Start of Frame (1 bit), ID (11 bits), Remote Transmission Request (1 bit), Identifier Extension (1 bit), Reserved (1 bit), Data Length Code (4 bits), Data/Payload (8 bytes), Cyclic Redundancy Check (16 bits), Acknowledgement (2 bits), End of Frame (7 bits). The appropriate fields are set to relay the occurrence of events on the bus.

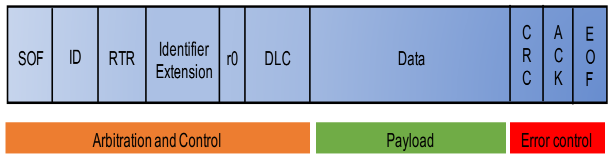


Fig. 2: Standard CAN frame structure.

Bus Control. When an ECU decides to transmit, it first needs to win control of the bus through an arbitration process where all ECUs transmit an identifier bit by bit. The combined state of the bus is the collective AND of all the transmitted bits at an instant. The transmission window is synchronized by means of an internal clock in every ECU. While contending, a lower value identifier always receives a higher priority to acquire control of the bus. If an ECU that transmits a recessive bit reads the state of the bus to be dominant, it withdraws from contention as a lower value ID is active. As a result, one ECU will win the arbitration process at the end. Once control has been acquired, an ECU transmits the rest of the frame in order. The receiving ECUs can either choose to accept or discard the transmitted frame based on the acceptance mask set on the node. The acceptance mask allows an ECU to filter all the transmitted frames and accept those that are relevant to it. Later, every receiving node sends an acknowledgment bit to the sender after transmission to indicate proper reception. If an error is noticed by any receiving node, a series of six bits, also known as an error frame, is sent to destroy the content of the transmitted frame. The nature of the error frame is based on the current error state of the node transmitting it. The error state of all ECUs is updated after a transmission cycle on the bus. Repeated errors can force an ECU into the bus off state where it refrains from transmitting as that could interfere with the normal operation of the bus.

Data Encoding. In this work, we focus on utilizing only the payloads in the frame for training our deep learning model.

A single payload from an ECU can have a maximum size of 8 bytes comprising one or more signals. As a result, the size of the data payload transmitted by an ECU is not fixed. An inherent difficulty in applying detection methods analyzing the payload field is due to the varying bit boundaries that define the number of bits needed for a signal. These boundaries are set by equipment manufacturers through custom rules that can vary widely between multiple vehicles. The traffic seen on the bus is therefore encoded and requires these rules from the manufacturer to extract individual signals. However, being data driven, NeuroCAN is capable of learning the structure of the data in any format and hence does not require any prior knowledge of the signals contained in the payload.

III. Related Works

In this section, we review several works that have employed deep learning for anomaly detection on CAN bus. In contrast to our work, these works do not utilize spatio-temporal dependencies in sensor readings and do not focus on detecting stealthy attacks at a frame level. Further, some of them operate with decoded CAN payloads, which limits their application.

LSTM-based Models. The authors in [2] modeled anomaly detection on the CAN bus as a supervised, binary and multi-label classification problem. The network employing LSTMs was built through a systematic process by experimenting with different number of layers, learning rates and activation functions. The obtained results indicate that the network is capable of identifying DoS and spoofing attacks with 100% accuracy while fuzzy attacks yield a few false positives and false negatives. In [13], the authors trained a model for detecting modified signals corresponding to the braking system. The network was later deployed as part of an in-vehicle Software-Defined Network to allow the detection module control data flows for taking appropriate steps after detection. A classification model with Convolutional LSTMs was presented in [11] for identifying DoS attacks. Further, one-shot learning was performed to train new models for identifying fuzzy attacks. In [10], a stacked LSTM predictor model was trained to detect the bit value in the data field of subsequent CAN frames. Additionally, the authors also propose five different methods for deriving an anomaly score over which the performance of the network is determined. The authors in [12] focused on predicting individual signal values such as vehicle speed with LSTMs. However, unlike [10], the implementation relies on decoded signal values as opposed to an unformatted bitstream.

Autoencoder-based Models. The authors in [14] proposed a deep contractive autoencoder focused on detecting fuzzy attacks. Models with different loss functions were developed and evaluated. In [1], a GRU-based autoencoder acting on individual signals in a CAN message was explored. A few other variants were also developed to analyze the importance of individual layers. In [15], the authors proposed a deep autoencoder comprising a single encoder and decoder structure. Additionally, the data was explicitly filtered to remove sequences of identifiers that are independent. Further, three models were developed with different reconstruction metrics and their performances were evaluated. In [16], a standard autoencoder with LSTM embeddings was proposed to identify flooding and continuous data change attacks. The authors in [17] developed a deep multi-task learning-based anomaly detection system where each task acted as a regularizer for the other. The architecture is composed of an LSTM encoder with two decoders: one working on reconstructing the input and the other on predicting the next maneuver of the vehicle.

Generic Network Models. In [18], the authors presented a detection model based on a feed forward neural network. Additionally, a feature extraction technique computing the probability distributions of every bit symbol was investigated. A multi-layer perceptron ensemble was employed in [19] for detecting data-content modifications in the CAN payload along with K-means clustering for preprocessing. A deep neural network with the triplet loss function was proposed in [20]. During training, the loss function focuses on minimizing the difference between the anchor and the positive sample. On evaluation, it was concluded that the network provides higher performance with increasing number of layers and loss functions other than the Softmax. An Intrusion Detection System based on Generative Adversarial Networks was proposed in [21]. The architecture comprised two different discriminators for the detection of well-known and zero-day attacks.

IV. NeuroCAN Design

Overview. NeuroCAN is a deep learning model capable of learning the collective behavior of ECUs on a CAN bus. Although a set of standards have been employed in structuring a CAN frame, proprietary encodings and custom identifiers introduced by manufacturers make it cumbersome to design a single model capable of handling frames from multiple control units. Therefore a separate network is trained for every identifier. However, unlike [10] and [2] where the network is trained with data from a single CAN identifier, NeuroCAN architecture is designed to collectively operate on inputs from various sensors at an instant. As events inside a vehicle are often correlated, it is important to analyze the state of all ECUs when looking for stealthy attacks. Such attacks are more difficult to detect and can circumvent existing safeguards unless the observation in question is jointly evaluated with respect to spatio-temporal correlations among sensor readings on the bus.

Background on LSTMs. NeuroCAN employs LSTMs to learn spatio-temporal sensor correlations. The ability of feed forward neural networks is limited when applied for time series forecasting as the architecture lacks the means to remember the previous inputs. Recurrent Neural Networks (RNN) are better suited in this scenario as the output h_t at an instant t is dependent on the previous hidden state h_{t-1} as well as the current input x_t. However, while training, the incorporated feed-back design that introduces the previous state either causes gradients to shrink or grow rapidly and therefore RNNs struggle to learn long term dependencies.

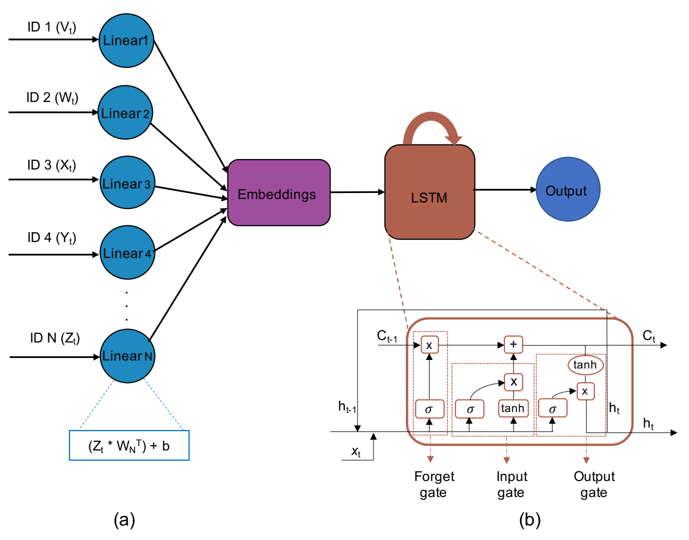


Fig. 3: NeuroCAN network structure.

LSTMs address this limitation by controlling the current state through three gates namely, 1) Forget gate 2) Input gate and 3) Output gate. These gates act on the current inputs and the previous cell state to explicitly regulate the magnitude of information that is retained across time. Fig. 3(b) depicts an architecture of an LSTM cell. The signals from the input and forget gate are given by

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{1}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{2}$$

where W, U are the weight matrices and b is the bias. The new cell state c_t is derived as a combination of c_{t-1}, i_t and f_t, where c_{t-1} is the previous cell state:

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t, \tag{3}$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c). \tag{4}$$

The output of the cell is then combined with current cell state to serve as input for the next time step.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{5}$$
$$h_t = o_t \tanh(c_t). \tag{6}$$

A. *System Architecture*

The high-level architecture of an anomaly detection system based on NeuroCAN is presented in Fig. 4. We consider a generic automotive CAN bus through which multiple control units relay sensor readings and other diagnostic data. These data points are generally consumed by other connected entities on the bus or a service technician attempting to pinpoint a fault in the vehicle. We assume that no hardware faults are present on the bus and the vehicle is not operated in abnormal environments or conditions that are unseen in its daily operation. Since our work focuses on identifying contextual anomalies, it is quintessential to determine a point in the network where data from every connection is visible. Therefore we consider deploying the trained models on the gateway as opposed to the distributed approach seen in [1]. Further, we assume that

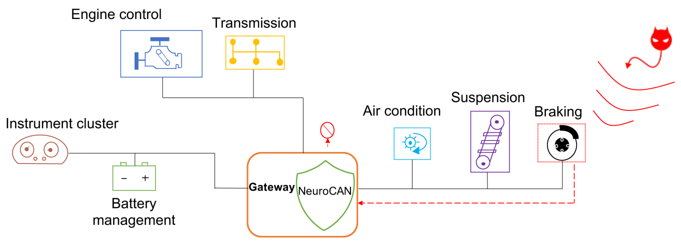


Fig. 4: System architecture: NeuroCAN runs on the CAN gateway.

the adversary has already compromised a connected ECU or has access to a device on the bus capable of inserting fabricated messages at specific instances. We do not discuss the techniques to gain access to the CAN bus as existing works [3], [22] have already proven its feasibility.

B. *NeuroCAN Network Structure*

NeuroCAN design comprises two fundamental components that are jointly trained with multidimensional time-series data. The structure of the NeuroCAN network is shown in Fig. 3. The initial section is an embedding layer following which is a single LSTM layer that acts on the derived embeddings to capture temporal relations in the data. Given that we train a network for every CAN ID, our model utilizes the current payload of the identifier and from other ECUs transmitting within the subsequent time step to predict the next payload sequence. Inputs from the other control units serve as additional context for NeuroCAN to infer the current state of an ECU relative to the collective behavior of other active ECUs on the bus.

Embedding Layer. As mentioned in Section II, the data field associated with each CAN ID varies depending on the transmitting ECU. Therefore, the data is first projected onto the embedding layer to obtain a fixed size vector for further processing. The embedding layer applies a linear transformation, shown in Fig. 3(a) over which `sigmoid` activation is applied to scale values between 0 and 1. The output from the individual embedding units is then cumulated and passed to the LSTM layer. During backpropagation, the weight matrices of only those emebedding units whose IDs were present in the current input sequence are updated. This ensures that a prediction given by the network is not dependent on other control units that did not transmit any frames during the current cycle. Our intuition is that since there exist multiple sensor readings within a single payload, developing individual embeddings for every identifier would allow the network to learn dependencies within an ECU while the LSTM layer focuses on capturing inter-ECU correlations.

LSTM Layer. This layer comprises a standard LSTM cell with `sigmoid` and `tanh` activations. The input to the LSTM layer is a scaled matrix of eight columns each representing a byte on the CAN payload and variable-size rows depending on the available context between two subsequent time steps. To elaborate, given an ID X, with n time sequences $\{X_1, X_2, X_3, ..., X_n\}$, the network predicts X_{t+1} given W_t, X_t, Y_t and Z_t at an instant t. Here, $\{t = 1, 2, 3, ..., n\}$ and W_t, Y_t and Z_t are data sequences

TABLE I: Dataset overview.

Dataset	Messages	Normal	Injected
Dataset 1	4,443,142	3,845,890	597,252
Dataset 2	4,621,702	3,966,805	654,897

TABLE II: Experimental results.

Dataset	Model	TPR	FPR	F1 score
Dataset 1	NeuroCAN	0.9	0.000065	0.95
	Baseline	0.89	0.23	0.83
Dataset 2	NeuroCAN	1	0	1
	Baseline	0.86	0.19	0.86

transmitted by other ECUs between the interval t and $t+1$. Further, $X_t = [x_{t1}, x_{t2}, x_{t3}, ..., x_{tk}]$ is an encoded vector of k sensor readings from an ECU at time step t. Whenever a set of sequences is passed through the network, each payload is first forwarded to its corresponding unit in the embedding layer. The output from the embedding network, described by $E_t = [E_{W_t}, E_{X_t}, E_{Y_t}, E_{Z_t}]$ is then passed to the LSTM layer and finally to the output layer with `sigmoid` activation. The output layer also applies a linear transformation similar to the embedding layer and calibrates the final output. The predicted result is then compared against the expected sequence and the loss is computed using the Mean Squared Error (MSE) as

$$\text{MSE}(X, \hat{X}) = 1/k \sum_{i=1}^{k} (x_i - \hat{x_i})^2 \,. \qquad (7)$$

As a result, the individual byte losses are squared and averaged over the entire sequence. Finally, a signal is invoked if the computed anomaly score is greater than the determined threshold. We discuss the utilized approach for threshold estimation in the next section. The final network architecture was selected through repeated trials, experimenting with different number of LSTM units and activation functions.

V. Experiments and Results

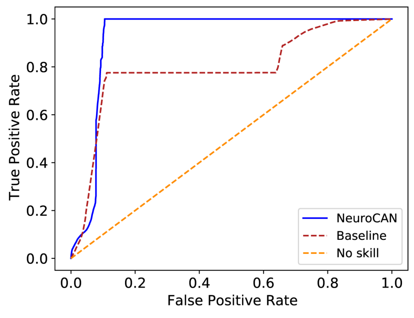
In this section, we first describe our experimental setup and provide an overview of the utilized datasets. Later, we present our results and compare them against an existing baseline.

A. Baseline

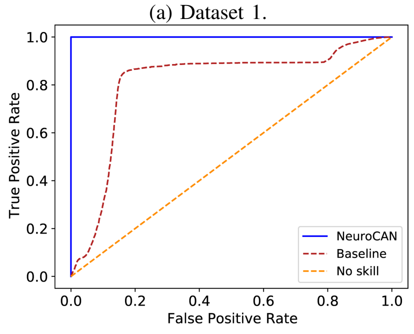
We establish the detection ability of NeuroCAN by comparing its performance against that of the network presented in [10] that serves as our baseline. Contrary to NeuroCAN, the baseline incorporates stacked LSTM layers with a higher number of hidden units. Unlike our approach, the payload from a CAN frame is first preprocessed into a binary matrix and is implemented over a rolling window. While training, the error produced by the network is computed with Binary Log Loss and an anomaly score is determined by taking the maximum of all losses in the given input. Further, a network for every CAN ID is trained with data only from that identifier and thus ECU correlations are not taken into consideration.

B. Training Setup and Parameters

For our experiments, we implemented and trained the models with `PyTorch` [23]. Additionally, we used `pandas` [24] and `scikit-learn` [25] for data preprocessing and evaluation. Training was performed on a Linux CentOS 8 machine, powered with an NVIDIA Tesla V100 and 32 gigabytes of memory. We trained our model with data sequences only from normal CAN frames. The final hyperparameters were chosen through repeated experimentation and testing. We applied mini-batch training with a batch size of one. The number of iterations was set to 50 epochs. Further, the Adam optimizer was utilized with a fixed learning rate of 0.001 [26].



(a) Dataset 1.

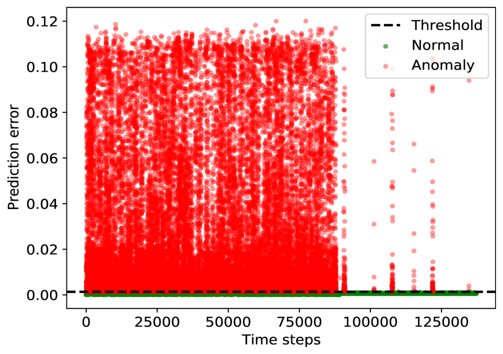


(b) Dataset 2.

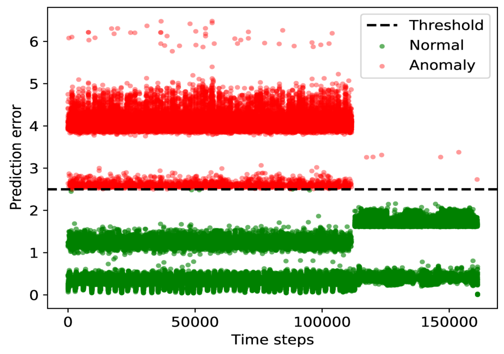
Fig. 5: RoC curves for different models.

C. Dataset Specification and Preprocessing

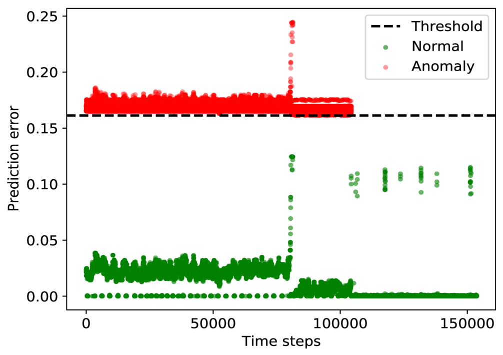
We evaluated our approach on two data sets obtained from the Hacking and Countermeasure Research Lab [21]. The datasets were developed by logging CAN traffic through the OBD port while injecting fabricated CAN messages. Specifically, messages related to the gear module and the RPM gauge were modified. For simplicity, we refer to these datasets as Dataset 1 and Dataset 2 in the remainder of the paper. An overview of the number of samples in each dataset is given in Table 1. Each dataset comprises thirty to forty minutes of CAN traffic. The attack window varied between three to five seconds and an attack was performed for every millisecond during this period. A sample in a dataset comprised a timestamp, CAN identifier, DLC, data and a label to indicate the nature of the sample. As the dataset in its regular state was not adequate for training, a few preprocessing steps were applied. First, each dataset was split with 70 percent allocated for training and the rest for testing. Further, all hexadecimal values were converted and normalized. Finally, the training and test sets were transformed by splitting the data field into eight features, each representing a byte on the CAN payload.



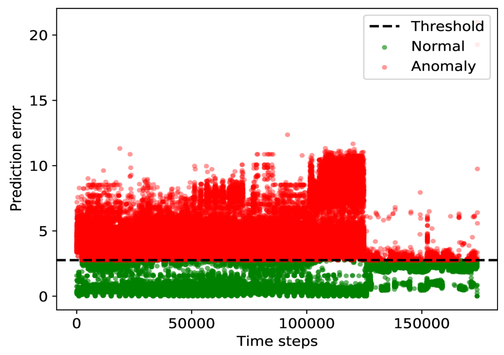
(a) NeuroCAN with Dataset 1.



(b) Baseline with Dataset 1.



(c) NeuroCAN with Dataset 2.



(d) Baseline with Dataset 2.

Fig. 6: Detection effectiveness for different models.

D. *Evaluation Metrics and Threshold Estimation*

The attack detection performance of the network was evaluated with the area under the Receiver Operating Characteristics (ROC) curve. The ROC curve is a plot of the false positive rate against the true positive rate under various thresholds. The selection of a reasonable threshold plays a critical role in the efficiency of NeuroCAN. We determined an optimal threshold for each model through enumeration of F1-scores, given by

$$F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \qquad (8)$$

The F1-score is a weighted average of the precision and recall and is utilized to estimate the machine learning performance when there exists an imbalance in the predicted classes. Ranging between 0 and 1, a score closer to one indicates a stronger model. Compared to other heuristic-based threshold moving approaches, we found that determining the threshold with the F1-score resulted in fewer false positives.

E. *Results and Discussion*

Attack Detection. The true positive rate, false positive rate and F1 scores of NeuroCAN and the baseline for both datasets are summarized in Table II. We specifically turn our focus to the true and false positive rates as the cost of misclassifying an anomalous sequence is much higher and possess severe consequences. The ROC curves in Fig. 5(a) and Fig. 5(b) compare the detection performance of the model and the baseline when evaluated with each dataset. As seen in the plots, NeuroCAN considerably outperforms the baseline in all cases. Furthermore, with Dataset 2, the trained network results in a perfect classifier capable of detecting all anomalies in the test set with zero false positives and false negatives. Also, we observed that the model misclassified a small fraction of normal sequences as anomalies in Dataset 1. This can be attributed to the imbalance in the training set caused by the nature of the payload on the CAN bus as not all sequences are transmitted at the same frequency.

Hyperparameter Tuning. In this experiment, we studied the variation in the performance of NeuroCAN by varying the size of the hidden state in the LSTM layer. Specifically, we evaluated the model by setting the number of hidden units to 8, 16, 32 and 64. As seen in Table III, there is no change in performance with the model trained on Dataset 2 in any case. With Dataset 1, we observed that the model with 8 hidden units showed higher classification performance when compared to hidden sizes of 32 and 64. However, it incurred the highest number of false positives while the others yielded significantly lower numbers.

Model Efficacy. The effectiveness of a deployed anomaly detection system is mainly dependent on its ability to establish a clear boundary to identify anomalies and the time it takes for detection as performing evasive maneuvers is time critical when the vehicle is in operation. As seen in Fig. 6(a) and Fig. 6(c), NeuroCAN establishes a clear boundary to distinguish the abnormal payloads. With Dataset 1, we observed a larger margin of difference resulting in the prediction error being very close to zero for normal CAN traces. This validates the ability of our system to learn the structure of the encoded data field and as such leads to higher prediction errors for the injected payloads. Further, we observed a smaller margin

TABLE III: Results of hyperparameter tuning.

Dataset	Hidden units	TPR	FPR	F1 score
Dataset 1	8	0.92	0.015	0.951
	16	1.0	0.0	0.947
	32	0.9	0.000065	0.951
	64	0.9	0.00049	0.948
Dataset 2	8	1.0	0.0	1.0
	16	1.0	0.0	1.0
	32	1.0	0.0	1.0
	64	1.0	0.0	1.0

of difference in the baseline models, shown in Fig. 6(b) and Fig. 6(d) resulting in more misclassifications. During the detection phase, we also noticed that our model took close to half the time taken by the baseline to provide an inference. In particular, NeuroCAN took 1.7 milliseconds on average while the baseline took 3 milliseconds to predict the next sequence.

Effect of Context. To further comprehend the impact of the pairwise correlations present between multiple ECUs, we modified the NeuroCAN structure by removing the embedding layer and trained the model with sequences from only the targeted identifiers. Though the model was able to detect most of the injected samples, we observed a slight decrease in performance with increased number of false negatives. Fig. 7 illustrates the effect of sensor context shown by the variation in detection accuracy.

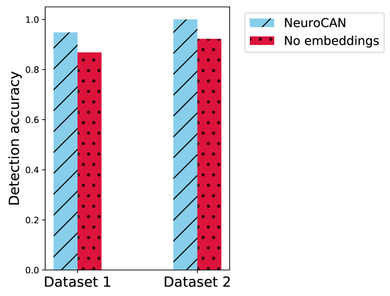


Fig. 7: Impact of sensor context on detection accuracy.

VI. Conclusion

With the ongoing rise of autonomous vehicles, safeguarding on-board systems such as the CAN bus is a growing concern for vehicle manufacturers. In this paper, we propose NeuroCAN, a deep learning-based system to primarily detect contextual anomalies caused by stealthy attacks in the CAN bus. The results indicate that NeuroCAN achieves a low false-positive rate and can identify such attacks with 95% accuracy. Furthermore, NeuroCAN performs considerably better than the studied baseline. However, the current model has not been subjected to retraining over time. As a result, the system might fail to be in sync with the transmitted parameters that can change due to replaced parts, wear and tear etc. Therefore it is imperative to incorporate online training which entails our future work.

References

[1] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems," *arXiv preprint arXiv:2007.08795*, 2020.

[2] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based Intrusion Detection System for in-vehicle CAN bus communications," *IEEE Access*, vol. 8, 2020.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Security Symposium*, vol. 4, 2011.

[4] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus," in *Proc. ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

[5] E. Wang, W. Xu, S. Sastry, S. Liu, and K. Zeng, "Hardware module-based message authentication in intra-vehicle networks," in *Proc. ACM/IEEE ICCPS*, 2017.

[6] R. Bosch *et al.*, "CAN specification version 2.0," *Robert Bousch GmbH, Postfach*, vol. 300240, 1991.

[7] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, "Automotive intrusion detection based on constant CAN message frequencies across vehicle driving modes," in *Proc. ACM Workshop on Automotive Cyber-security*, 2019.

[8] K. T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for vehicle intrusion detection," in *Proc. USENIX Security Symposium*, Aug. 2016.

[9] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intelligent Vehicles Symposium*, 2011.

[10] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with Long Short-Term Memory Networks," in *Proc. IEEE DSAA*, 2016.

[11] S. Tariq, S. Lee, and S. S. Woo, "CANTransfer: transfer learning based intrusion detection on a Controller Area Network using convolutional LSTM network," in *Proc. Annual ACM Symposium on Applied Computing*, 2020.

[12] M. Abbas, M. Safar, and A. Salem, "Anomaly detection system for altered signal values within the intra-vehicle network," in *Proc. IEEE DTIS*, 2020.

[13] Z. Khan, M. Chowdhury, M. Islam, C.-Y. Huang, and M. Rahman, "Long Short-Term Memory neural networks for false information attack detection in Software-Defined in-vehicle network," *arXiv*, 2019.

[14] S. F. Lokman, A. T. Othman, S. Musa, and M. H. A. Bakar, "Deep contractive autoencoder-based anomaly detection for in-vehicle Controller Area Network," in *Progress in Engineering Technology*. Springer, 2019.

[15] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," in *Proc. ACM/IEEE DAC*, 2020.

[16] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: an unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, 2020.

[17] V. Sadhu, T. Misu, and D. Pompili, "Deep multi-task learning for anomalous driving detection using CAN bus scalar sensor data," *arXiv preprint arXiv:1907.00749*, 2019.

[18] M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PloS one*, vol. 11, no. 6, 2016.

[19] S. Boumiza and R. Braham, "An anomaly detector for CAN bus networks in autonomous cars based on neural networks," in *Proc. IEEE WiMob*, 2019.

[20] A. Zhou, Z. Li, and Y. Shen, "Anomaly detection of CAN bus messages using a neural network for autonomous vehicles," *Applied Sciences*, vol. 9, no. 15, 2019.

[21] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. Annual Conference on Privacy, Security and Trust*, 2018.

[22] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019.

[24] The pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[25] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.