CANintelliIDS: Detecting In-Vehicle Intrusion Attacks on a Controller Area Network Using CNN and Attention-Based GRU

Abdul Rehman Javed, Saif ur Rehman, Mohib Ullah Khan, Mamoun Alazab, *Senior Member, IEEE*, and Thippa Reddy G

***Abstract*—Controller area network (CAN) is a communication protocol that provides reliable and productive transmission between in-vehicle nodes continuously. CAN bus protocol is broadly utilized standard channel to deliver sequential communications between electronic control units (ECUs) due to simple and reliable in-vehicle communication. Existing studies report how easily an attack can be performed on the CAN bus of in-vehicle due to weak security mechanisms that could lead to system malfunctions. Hence the security of communications inside a vehicle is a latent problem. In this paper, we propose a novel approach named CANintelliIDS, for vehicle intrusion attack detection on the CAN bus. CANintelliIDS is based on a combination of convolutional neural network (CNN) and attention-based gated recurrent unit (GRU) model to detect single intrusion attacks as well as mixed intrusion attacks on a CAN bus. The proposed CANintelliIDS model is evaluated extensively and it achieved a performance gain of 10.79% on test intrusion attacks over existing approaches.**

***Index Terms*—Controller area network, cyberattacks, in - vehicle network (IVN) security, intrusion detection, security protocols.**

I. Introduction

CAN is a sequential network technology that has been widely deployed in the in-vehicle network (IVN) for more than three decades to provide an effective, stable, economic, and reliable broadcast communication between electronic control units (ECUs) [1], [2]. CAN allows several electronic units on a vehicle to communicate essential control data. This technology has been effectively applied in numerous industrial areas, for example, process control and assembling, x-ray machines, aircraft, railway, and textile machines [3], [4]. Due to high adaptability and simple structure, communication in CAN bus can be under cyberattacks [5], [6]. Song *et al.* [5] and Lee *et al.* [6] demonstrate that the receiving node does not verify the source of a CAN message. Hence, it may lead to many network traffic injection attacks. The three primary susceptibilities of in-vehicle CAN are weak access control, no confirmation, and no encryption. Lokman *et al.* stated the criticality of cybersecurity in CAN by stating the non-presence of authentication mechanism [7]. Broadcasting of CAN bus makes it difficult to get control, for example, any malicious/ anomalous ECU joined to the CAN Bus can communicate outlines. So the security of the CAN bus depends on legitimate verification and encryption [8]. Initially, Hoppe *et al.* [9], used the black box to perform basic attacks on the gateway ECU. They show how an attacker can sniff communication on the CAN bus. Invaders can manipulate and spoof priority bits in the CAN bus because all nodes can access single bus [10]. Boudguiga *et al.* [11] reported in their findings that weak security mechanisms of CAN protocol make it a critical target for Fuzzy, spoofing, DOS, impersonation attacks. These attacks appear to be real viewing traffic sequences, and it is hard to recognize them from the normal sequence.

To provide security and detect attacks at the earliest, in this approach, we make the following contributions:

- Novel approach: We developed *CANintelliIDS*, a novel generalized intrusion detection approach based on convolutional attention incorporated with gated recurrent neural network (GRU) named *CANintelliIDS*, that is capable of detecting intrusion attack in a CAN bus.
- Single intrusion detection: *CANintelliIDS* analyzes the data sequence without ground truth to detect and identify each single intrusion attack.
- Mixed intrusion detection: *CANintelliIDS* analyzes the data sequence in the form of custom vectors with ground truth to detect and identify mixed intrusion attacks.
- Experimentation and evaluation: We experimented *CANintelliIDS* on the CAN dataset collected from real vehicles and evaluated the accuracy when the dataset is balanced, accuracy and F1-score when the dataset is imbalanced. We provide a comparison of *CANintelliIDS*

Manuscript received January 10, 2021; accepted February 13, 2021. Date of publication February 19, 2021; date of current version July 7, 2021. Recommended for acceptance by Dr. Liang Zhao *(Corresponding author: Mamoun Alazab.)*

Abdul Rehman Javed is with the Department of Cyber Security, Air University, Islamabad 44230, Pakistan (e-mail: abdulrehman.cs@au.edu.pk).

Saif ur Rehman is with the Faculty of Computing and AI, Air University, Islamabad 44230, Pakistan (e-mail: 181065@students.au.edu.pk).

Mohib Ullah Khan is with the National University of Computer and Emerging Sciences, Islamabad 47040, Pakistan (e-mail: mohib_khn@outlook.com).

Mamoun Alazab is with the College of Engineering, IT, and Environment, Charles Darwin University, Casuarina, Northern Territory NT 0815, Australia (e-mail: alazabm@ieee.org).

Thippa Reddy G is with the School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India (e-mail: thippareddyg@vit.ac.in).

Digital Object Identifier 10.1109/TNSE.2021.3059881

with state-of-the-art studies and baseline approaches and report that we achieve higher attack detection performance for both training and testing data.

- Effectiveness of the proposed approach: We tune our *CANintelliIDS* model, by keeping in mind the time complexity and best performance. For one message of the real-world CAN bus data, our model is proficient in detecting the data sequence as normal or intrusion attack through experimentation at the earliest with high accuracy.

The rest of the paper is systematized as follows. Section II provides a brief analysis of the current literature in CAN protocol, authentication approaches, various attacks, and intrusion detection approach. Section III demonstrate the data overview. Section IV provides the problem formulation and our proposed methodology for intrusion detection on a CAN bus. Section V shows our experiments and results. Section VI offers observations, limitations, and discussion. Finally, Section VII concludes the paper and discusses future work.

II. Related Work

CAN is a pathway of information sharing between nodes in vehicles, thus, a hacker can inject misleading information into the network and can harm the system. Lee *et al.* [6] suggested detection of intrusion in the system based on the investigation of the counterbalance ratio and time intermission between request and response communications in CAN. In case a remote frame has a specific classification, then the receiver node ought to reply to the remote frame instantly. In normal working order, every individual node has a fixed response counterbalance ratio and time intermission whereas there is variation observed in the attacking phase. Young *et al.* investigate the remarkably productive intrusion detection approaches for securing vehicular networks. Their study's main focus is mainly based on the CAN. They delivered an explanation of vulnerabilities, discussed threat models, highlighted current attacks on CAN, and debate the pros and cons of recommended explanations [12].

Dupont *et al.* proposed a methodology to perform a comparison of CAN network intrusion detection systems (NIDSs) on balanced criteria and assessed with the others. They identified the limitation in the literature, they mentioned and highlighted the reasons for difficulties in differentiating CAN payload from benign to malicious. They devised their novel approach named "meaning-aware" detection of CAN security [13]. W. Choi proposed a novel identification technique, which operates the physical layer of the vehicle internal CAN. By utilizing unique features of electrical CAN signals, their system detects malicious ECUs. Their system is designed by introducing an additional monitoring unit to the present network. This property allows their system to be embedded in current compliance with required CAN standards. Their system provided a remarkable accuracy with an uncertainty percentage of 0.36% on average [14]. Tang *et al.* presented a deep learning-based approach in the software-defined networking (SDN) architecture and a gated recurrent neural network (GRU-RNN). They evaluated their model over the NSL-KDD dataset [15]. Guo *et al.* presented a deep convolutional neural network incorporated with long short-term memory (LSTM), multi-layer perception, and the attention mechanism in an end-to-end model to detect the intrusion in the network [16].

Tariq *et al.* developed an interruption discovery algorithm to detect DoS, replay, and fuzzy attacks inserted in a real-life test environment vehicle. They utilized heuristics and Recurrent Neural Networks (RNNs) to identify attacks. They evaluated their algorithm gathered from KIA Soul. Their Algorithm showed promising results on that particular dataset [10]. Wu *et al.* introduced an in-vehicle networks (IVNs) environment, and the limitations and features of an intrusion detection system (IDS) strategy for IVNs. They surveyed the introduced IDS designs for the IVNs and highlighted the consistent disadvantages. Several optimization purposes were measured and extensively equated. They discussed the tendency, current problems, and evolving research dimensions as well [17]. Taylor *et al.* created LSTM based neural network that is capable of learning to predict the next data sequence initializing from each sender of the data on the bus. By synthesizing anomalies with modified CAN bus data, their model successfully flags the suspected anomalies [18]. Tariq *et al.* proposed an approach "CANTransfer," an interruption discovery system using Transfer Learning for CAN bus. They trained a Convolutional LSTM based model by utilizing acknowledged intrusion to identify test attacks. They also applied one-shot learning so the system can detect test intrusions attacks adaptively variable datasets [19]. Moore *et al.* highlighted the requirement for making CAN bus safer by distinguishing irregular traffic outlines via infrequent refresh rates of a particular command. They introduced an algorithm with experimentations using three attacks in five testing environments. Their algorithm took only 5 seconds to distinguish malicious inputs and achieved true benign/malicious detection rates of 0.9998/0.00 298, respectively [20]. Buttigieg *et al.* contributed to CAN security by analyzing and testing the CAN current security measures and evaluated the strength of CAN protocol by performing tests on a BMW E90 (3-series) instrument cluster. The examination was performed by developing a device locally which they connected to the same CAN-Bus as hijack attack the device to send deceived inputs to the instrument cluster [21].

Choi *et al.* proposed a novel approach concentrating on fortifying an in-vehicle CAN network named as automotive intrusion detection system (so-called VoltageIDS). Their model influences the unique features of an electrical CAN signal as a pattern of the electronic control units (ECUs). The remarkable contributions that the model they proposed is that it didn't require modification in current systems and they examined their model on the real testing environment on actual vehicles being on the road [22]. Liu reviewed the weaknesses of in-vehicle networks comprehensively and explained the basic concepts, and summarized the attacking techniques and their procedure. They also suggested the solutions to those attacks and future vision [23]. Olufowobi proposed an algorithm to explore the real-time model of the CAN and developed a feature-based intrusion detection system (IDS) by utilizing overseen learning with the present-time model as an input

TABLE I
CHARACTERISTICS OF THE CAN DATASET FOR INTRUSION DETECTION (OTIDS)

Characteristic	Value
Timestamp	Recorded time (s)
DLC	Data bytes quantity, from 0 to 8
CAN ID	Identifier of CAN information in HEX (ex. 043f)
DLC	Data bytes quantity, from 0 to 8

feature. Their performance evaluations were held on a sedan car examination [24]. Zhewen *et al.* introduced a novel sequence-to-sequence model by utilizing the Attention-based Gated Recurrent Unit (AGRU). Its main purpose is to improve the accuracy of forecasting procedures. It implants the job of correlating different forecasting step by hidden activations of GRU blocks. Besides, an attention procedure is intended as a feature selection technique to classify the important input variables [25]. Song et. al [26] introduced a deep convolutional neural network while reducing the unnecessary complexity in the architecture of the Inception-ResNet model to achieve higher accuracy while detecting intrusion attacks on the CAN bus.

To our best knowledge, there exist several research gaps that are apparent in the exiting work focusing on CANs. To start with, as far as we could know, existing studies lack deep learning implementations in intrusion detection for CANs. Normal CAN network data is getting immensely complex and deep learning models are the optimal choice for performing proficiently over large datasets. In most cases, deep learning models outperform conventional intrusion detection and identification models. Furthermore, existing studies lack in providing ensemble intrusion detection models that increase the performance and decrease the weakness of the individual intrusion detection techniques in CANs. Existing studies also lack in performance when intrusion data is available in small magnitude. Existing studies also lack in the identification of intrusion types (i.e., DDOS, fuzzy, and impersonation). Finally, applications concentrating on real-time detection of intrusion data are limited.

III. DATA SELECTION AND ANALYSIS

The dataset which we used to test our system is initially published by [6]. This dataset includes fuzzy attack, DoS attack, impersonation attack, and attacks-free states. The Datasets were created by categorization of CAN traffic via the OBD-II port from an actual automobile while message inserting attacks were performing. Lee *et al.* used KIA SOUL (subcompact crossover SUV) [6]. The following are the type of attacks: a) DOS Attack b) Fuzzy Attack. c) Impersonation Attack. Table I shows the characteristics of CAN dataset for Intrusion Detection (OTIDS) [6]. The timestamp graphs show the graphical representation of the data insertion in a CAN bus during a certain event.

Figure 1 demonstrates the architecture of the CAN-bus frame which is composed of seven sections. The explanation of the fields in the CAN-bus frame is as follows:

- Start of frame: Initialize the bus when in an idle state.
- Arbitration Field: It comprises 11 bits as an identity and the 12th bit as RTR (Remote Transmission Request). The identity bits are used as a priority while the arbitration procedure and the RTR are identified conferring to the class of CAN frame.
- Control Field: It designates two bits held in reserve and a Data Length Code (DLC) of four bits.
- Data Field: It denotes the actual data information that is being transferred to some other node. It contains a minimum value of 0 and a maximum value of 8 bytes.
- CRC Field: It ensures the legitimacy of information as Cyclic Redundancy Code (CRC).
- Acknowledge Field: It comprises the ACK part and ACK delimiter part, both denoted by a single bit.
- End of Frame: To terminate the CAN frame, a check containing seven recessive bits is used.

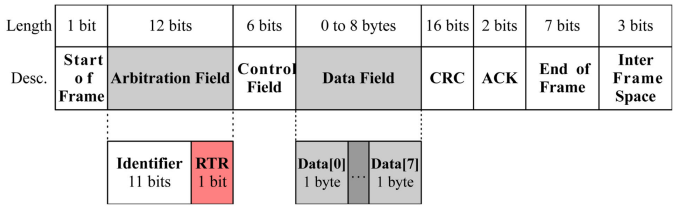


Fig. 1. Format of CAN Bus Frame [6].

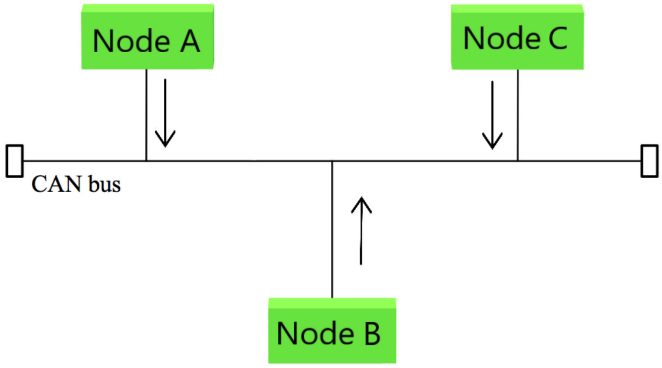


Fig. 2. Typical Overview of Attack-Free State in a CAN Bus.

A. Attack Free Communication

Attack free state depicts the normal functionality of an information transfer through CAN bus between nodes in Figure 2. This state of the CAN-Bus has all nodes green which shows that all nodes transmit good signals. The information from all nodes is benign and it travels along CAN-bus without any interruption. The dataset contains 2 369 868 Attack-free state messages. This Architecture further modifies in upcoming subsections to show the changes that occur when it is affected by any attack. Different attacks try to interrupt the communication between nodes and corrupt the information which they share.

Figure 3 shows the timestamp graph of CAN in the attack-free state. The graph shows a uniform flow of data/information and thus it is clear that no attack is occurring. The variation in timestamp is consistent which shows a flow of normal signal through the bus.

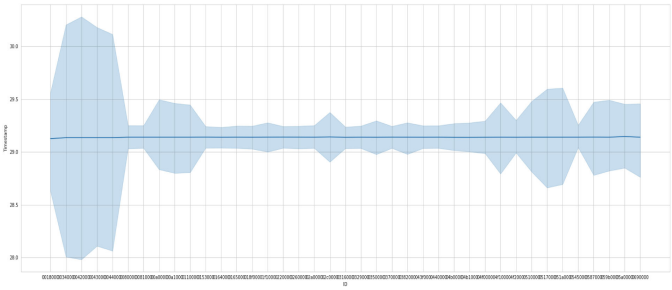


Fig. 3. Time Sequence Graph to Analyze Attack-Free Traffic in CAN Bus where no attack is occurring.

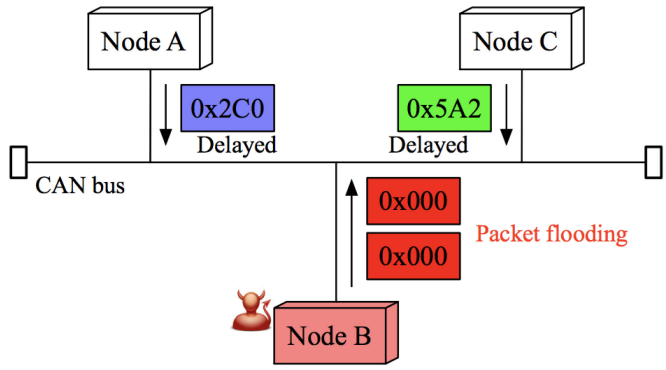


Fig. 4. Typical Overview of DoS Attacks in a CAN Bus.

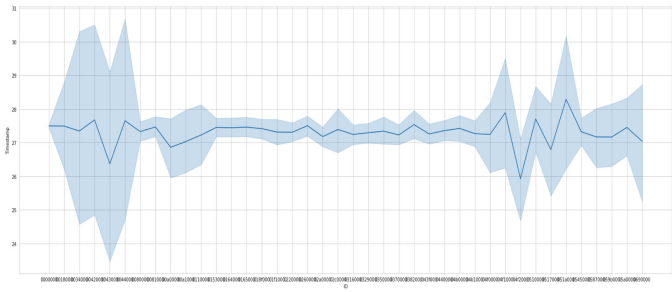


Fig. 5. Time Sequence Graph to Analyze DoS Attack in CAN Bus Readings.

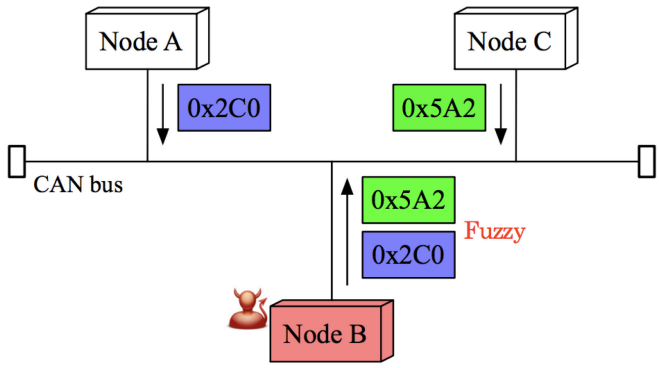


Fig. 6. Typical Overview of Fuzzy Attacks in a CAN Bus.

B. DOS Attacks

A denial-of-service (DoS) attack is a kind of cyber-attack hacker's intention is to do the malicious activity by rendering a computer server or any hosting device to make it unavailable for its users by interrupting the device's standard functionality [27], [28]. The methodology of DoS attacks is to flood a targeted service provider with requests such that true users may not be able to use it anymore. The dataset contains 656 579 number of DoS attack messages. Figure 4 depicts that Node A transmits a message and so as Node C but the transmission flow is delayed because of false packet flooding from Node C [6]. Node C transmits the large number of packets containing null information. Thus whole CAN bus becomes unavailable for useful information flow.

Figure 5 shows the behavior of CAN bus information transfer during a DoS attack using a time sequence graph. The graph shows absurdity as the null information packets are being transmitted by Node C. This flood transmission of null packets cause high deviations in peaks and their regularity. This plot shows an anomaly in data flow due to a DoS attack is occurring.

C. Fuzzy Attacks

A fuzzy Attack is a sort of cyberattack in which a hacker injects a massive volume of packets into the network using anonymous and randomly created packet IDs and delays access to other nodes. The dataset contains 591 990 number of Fuzzy Attack messages. A hacker can insert information of randomly spoofed identifiers with random data. As a result, all

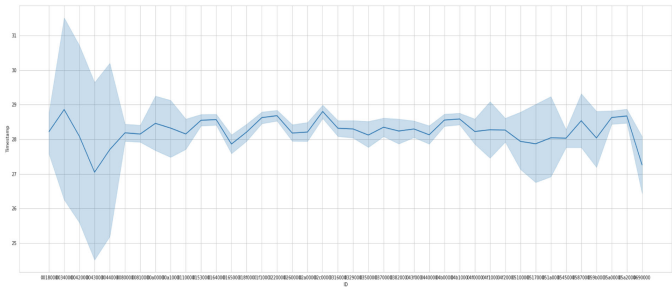


Fig. 7. Time Sequence Graph to Analyze of Fuzzy Attack CAN Bus Readings.

nodes receive lots of functional messages, and it causes unintended vehicle activities. To make use of the fuzzy attack, a hacker analyses in-vehicle information massages, and selected target identifiers to make the system react with unpredicted actions. When Node C injects massive packets of the same identity having null information into the CAN-bus, the original packets which are released by Node A and Node B respectively are delayed to reach their destination as Node C is overloading CAN-bus as shown in Figure 6 [6].

Figure 7 show the behavior of CAN bus information transfer during a Fuzzy attack using a time sequence graph. The graph shows absurdity by showing in delay as compared to the Attack-free state in Figure 3. Node C releases the flooding packets of false information. This flood transmission of null packets causes high deviations in peaks and their regularity from the mean and thus causes a delay in the original packet reaching its destination. This plot shows an anomaly in data flow due to fuzzy attacks is occurring.

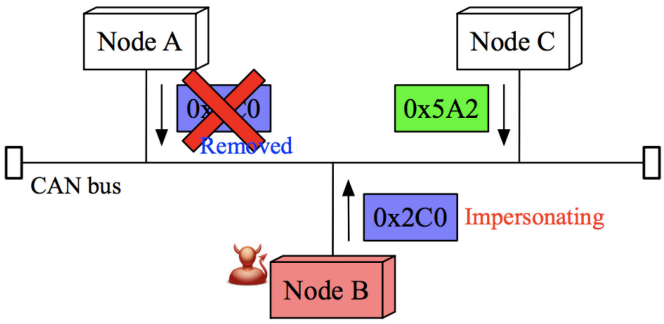


Fig. 8. Typical Overview of Impersonation Attacks in a CAN bus.

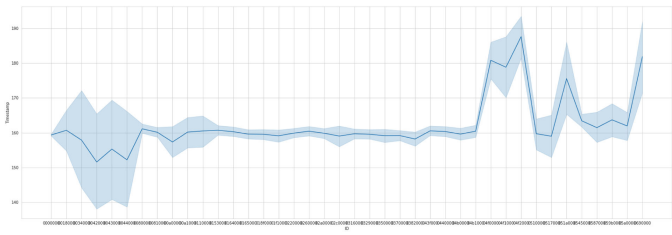


Fig. 9. Time Sequence Graph to Analyze of Impersonation Attack CAN Bus Readings.

D. Impersonation Attacks

An impersonation attack is a sort of cyberattack in which a hacker successfully captures the identity of an authentic entity in a system or a communications protocol. The dataset contains 995 472 Impersonation Attack messages. In Figure 8, Node C captures the packet information of Node A. After adopting the identity of Node A's packet, it transmits the false packet with the same identity of the packet that Nodes A's packet contains and removes the benign information packet which is originally transmitted by Node A [6]. Thus when Node C transmits information in CAN bus, it is identified as benign but it contains malicious information resulting bad impact on the system.

In Figure 9, the time sequence graph shows the behavior of the system when facing an impersonation attack. The attack deletes the original packet and deviates the lower than normal and due to this malicious packet injection, the outcome becomes abrupt changes in peak since the packet contains malicious information but having the identity of the original packet. This plot shows an anomaly in data flow due to impersonation attacks is occurring. Moreover, the proficiency of the model is evident in model performance such that it analyzes the attack unsupervised.

Table II shows the selected dataset to detect the intrusion attacks for this work. Data1 to data8 represents the data points that are formed in a data sequence used to analyze the attacks. We converted this data sequence into vectors and then feed this data to our *CANintelliIDS* model for intrusion attack detection and identification.

IV. Methods

In this section, we firstly discuss the two models that form the building blocks of our approach namely the CNN model and attention-based GRU developed in combination to detect single intrusion attacks and mixed intrusion attacks on a CAN bus. The main difference between single and mixed intrusions is that single intrusion is denoted when there is only one sort of attack being executed and mixed intrusion is denoted when multiple sorts of anomalies are found in the data stream of the CAN bus [29], [30]. Initially, the shape of the CAN dataset is in Hexa numeric points so we change it into a data sequence (e.g., data1 to data8 is our complete sequence for identifying the intrusion attack) as shown in Table II. The timestamp is transformed into a time difference, which specifies the time interval between the present and preceding packet.

Computing and Model Configuration: The PC specifications include Intel(R) Core(TM) i7-7700HQ CPU 8th Generation @ 2.80 GHz, NVIDIA GeForce GTX 1060Ti GPU and Linux 20.04 for the operating system. For developing the algorithm, we used Python 3.7 with Tensorflow 2.0 [31] Use Scratch Code for attention mechanism and deep learning model.

A. Convolutional Neural Network (CNN)

In our experiments, CNN plays the role of a neural network that has one or more than one convolutional layers and is mainly utilized for real-time detection and classification of inconsistent sensor behavior and to automatically detects the important features also for other autocorrelated data [32]. The core benefit of CNN compared to its precursors is computationally efficient and detects hidden pattern sequences in intrusion attacks extremely well. The input to CNN is a series of data sequences from the continuous feed of CAN Bus. CNN, therefore, utilizes the data from multiple CAN nodes simultaneously for the detection and identification of anomalous patterns. The proposed model consists of two convolution layers with max-pooling, followed by two fully connected layers with a random dropout of 0.3 between the layers. A 2×2 pool size and 40, 60, and 60 filters size are used for convolutional layers respectively. It is well known that the CNN model tends to overfit during the training of known data. To overcome this limitation, we applied weight Constraints on Layers using Keras that would be kernel_constraint, and bias_constraint with Maximum-norm it uses to force the learning weights to have a magnitude at or below a given limit which is 0.3.

B. Attention Based Gated Recurring Neural Network (AGRU)

AGRU consists of the reset gate and the update gate. Where reset gate controls the connection of new input with that of the old memory [33]. Whereas, the update gate determines the quantity of the precious memory that is necessary to keep around. To set the RNN model it is important to keep the reset gate to all $1's$ and update the gate on all $0's$. In AGRU the inputs sequence for both update and reset gates is given in the current timestamp input which is $x < t >$ and the hidden state of the previous timestamp is $C < t-1 >$. The output

TABLE II
REPRESENTATION OF THE INTRUSION DETECTION DATA SEQUENCE

Timestamp	ID	DLC	data1	data2	data3	data4	data5	data6	data7	data8
0.000271	0080000	8	00	17	dc	09	16	11	16	bb
0.0044201	0040000	8	00	16	ac	08	16	12	16	cc

is presented by a fully connected layer with a sigmoid activation function.

In a given timestamp t, the reset gate $\Gamma r \epsilon Rn \times H_s$ of hidden states and update gate $\Gamma u \epsilon Rn \times H_s$ of hidden states are computed as follows:

$$\Gamma r = \sigma(Wr[C < t-1 >, X < t >] + br) \quad (1)$$

$$\Gamma u = \sigma(Wu[C < t-1 >, X < t >] + bu) \quad (2)$$

Where, $(Wr, Wu)\epsilon Rd \times H_s$ of hidden states and Wr, Wu are weight parameters and br, bu are biases. We apply a sigmoid function to transform input values between (0,1).

In this step we integrate the Γr with a regular latent state updating mechanism. hidden state update equation be like:

$$\sim C < t >= tanh(Wc[\Gamma r * c < t-1 >, x < t >] + bc) \quad (3)$$

Where $\sim c < t >$ represents the candidate value which is the updated value, and Wc is a weight parameter and bc is a bias term.

Reset Gates represent how reverent $C < t-1 > 1$ to compute $\sim c < t > -1$ that is candidate value. we apply element-wise multiplication $C < t-1 >$ with Γr to reduce the previous state influence. So this rest gate is used to decide how much of the previous information to forget.

$$\sim C < t >= tanh(Wc[\Gamma r * c < t-1 >, x < t >] + bc) \quad (4)$$

Update Gates is required to find out the extent to which the new state $C < t >$ is just the old state $C < t-1$ and by how much the new candidate state $\sim C < t >$ is used. Therefore Γu is used for that purpose. If value of Γu is equal to 1 then $c < t > \approx \sim C < t >$ so if Γu close to be zero then it will never be suffer from vanishing gradient problem. The multiplication carried while computing $c < t >$ is element-wise multiplication

$$C < t >= \Gamma u * \sim C < t > +(1 - \Gamma u) * C < t > \quad (5)$$

So as mentioned if the value of Γu is close to 1, it simply retains its old state, and the information from $C < t >$ is essentially ignored, whenever Γu is close to 0, the new latent state.

To make the model more complex and efficient we apply an attention mechanism with that of the Recurring neural network for the identification of the Intrusion detection task. The main purpose of this method is to analyze anomalies within the system. CNN is mostly used to make the system more effective, fast, efficient, and to bring good significant features so those accurate predictions can be made with the help of the proposed model *CANintelliIDS*. To represent each data point within the data sequence and to combine CNN with that of *CANintelliIDS* so that target labels can be predicted. We use the attention mechanism as described below in equations [34].

$$atn_{ij} = softmax(f_{atn}(h_i, s_j)) \quad (6)$$

$$f_{atn}(h_i, s_j) = v_a \sim tanh([W_a h_i, W_a s_j]) \quad (7)$$

$$Contx_i = \sum_j atn_{ij} h_i \quad (8)$$

Where $Contx_i$ represents context vector, atn_i represents attention scores, h_i represents hidden state, s_j represents previous hidden state and $f_{atn}(h_i, s_j)$ represents attention function.

This mechanism works by generating the weights score of significant data points presented in 6 and 7. These weights scores are then multiplied with their respective data point vectors. The results are then added up to form a single context vector as represent in 8. This context vectorC_i is then used for the prediction of the target label.

To detect the intrusion attacks in a CAN bus, a *AGRU* model is proposed to be used in combination with the CNN model explained previously in section IV-A. This model mainly consists of three bidirectional layers. *AGRU* comprises 250 memory units in the first hidden layer, a 150 memory unit is used in the second hidden layer and third layer respectively. The main advantage of GRU is it needs to train fewer parameters resulting in less memory consumption, faster execution and learns efficiently from short data sequences. It deeply analyzes the data sequence based on the relational aspect and contextual information. The proposed *AGRU* model consists of 3 hidden layers, a random dropout value of 0.5 between the last 2 layers and apply weight decay constraint that is Maximum-norm (it use to force the learning weights to have a magnitude at or below a given limit) value of 0.3 to avoid the over-fitting.

C. CANintelliIDS Model

In this section, We elaborate on the proposed *CANintelliIDS* model, its proficiencies, and architecture. Figure 10 represents the attention mechanism of *CANintelliIDS*. To detect intrusion attacks, Firstly, data points are shown in Table II are converted into a vector sequence to be fed into the CNN layers. CNN extracts the significant feature from the input sequence to our *AGRU* model. *AGRU* learns the sequence based on the relational aspect and contextual information of the data points to tune its weights. The Attention mechanism assigns weights score to the data point sequence according to their significance as explained in 6. Next, it multiplies the weight score of the data points with corresponding vectors and generates a context vector 8 to predict the target label (i.e., DOS, fuzzy, impersonation, and attack_free). This model consists of three AGRU layers and two CNN layers. AGRU consists of (250 150,150)

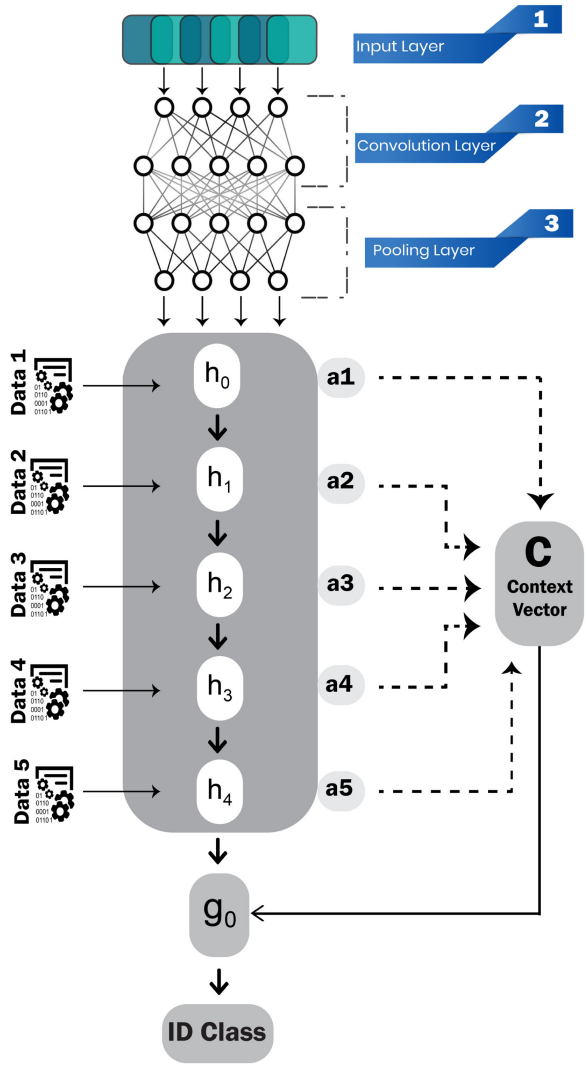


Fig. 10. A Graphical Representation of the Proposed CARGU Model to Detect and Classify Attacks to Increase Reliability of CAN Bus Transmission.

memory units in 3 corresponding layers while CNN consists of (128,64) neurons in 2 layers for the intrusion detection task. First, to train our custom w2v, *CANintelliIDS* convert all the data into vectors sequences. Next, it extracts 75% data sequences from the training set to train the model. To evaluate the performance for detecting intrusion attacks, 10% of sample data sentences are used as a validation set and 15% for the testing set.

The performance of the model can be affected by many hyper-parameters in the *CANintelliIDS* network. Hyper-parameter optimization or tuning is the issue of choosing a group of optimal hyper-parameters for a learning algorithm. Hyper-parameters are significantly important as they directly regulate the behavior of the training algorithm and have a notable influence on the performance metrics of the model. To overcome this problem, we tune parameters to make our model more efficient. To avoid over-fitting we apply the weight decay method of L2-regularization with the value 0.0003, dropout with a probability of 20, and Relu (non-linear) activation function between layers. For training, we use 200

Algorithm: 1 Multi-Stage Attention Mechanism With a Convolutional AGRU (CNN-AGRU)

Input: Intrusion Attack Sequence
Output: Attack Prediction

Pre-processing (Intrusion data sequence)
$X \leftarrow Adata$
$T \leftarrow H(L)$ # (One-Hot Encoding)
$\mu \leftarrow \frac{1}{m} * \sum_{i=1} X^{[i]}$ # (Normalize Data)
$X \leftarrow X^{[i]} - \mu$
$\sigma^2 \leftarrow \frac{1}{m} * \sum_{i=1} X^{[i]2}$
$X^{[i]} / = \sigma^2$
$D_2 \leftarrow Array(X^{[i]})$ # (Matrix Convergence)
for j *in* $range(len(L_y)$ **do**
    $W^{[j]} \leftarrow rand(m \times n) * \sqrt{\frac{2}{n[j-1]}}$
end for
$\mathbf{CNN-AGRU}(D_2)$
$D_3 \leftarrow \mathbf{RM}(D_2)$ # (2D to 3D Shape)
$F_m \leftarrow \mathbf{CNN}(D_3)$ # (CNN Model)
$V_s \leftarrow \mathbf{MaxPooling}(F_m)$ # (Feature-map vectors)
$R(ac) \leftarrow \mathbf{GRU}(V_s)$ # (Attention GRU)
$S_w \leftarrow \mathbf{Attenion}(R(ac))$ # (Scores)
$C_xV \leftarrow \mathbf{Context}(\sum (S_w \times S_v))$
$TL \leftarrow \mathbf{Predict-Class}(C_xV)$
return TL

epochs with a batch size of 120 and to speed up the process. Adam(Stochastic) optimizer is used with the decay rate is 0.035. To generalized *CANintelliIDS* model, we apply model checkpoints to record all the network weights and loss is detected after the execution of each epoch. We select the best set of weights to instantiate our *CANintelliIDS* model. Softmax activation is deployed at the output layer to predicts the Target labels. These are parameters that are tuned to detect the intrusion attacks efficiently.

D. CNN-AGRU Model

Algorithm 1 represents the demonstration of our proposed model *CNN-AGRU*. Where X represents the anomaly data$Adata$ and m shows the number of samples. H is the one hot transformer function used to convert the target-labels L into vectors(contains 0-1) T. The next phase is the normalization of data, the first step is to take the mean and then subtract the mean μ from X after that normalized variance σ is applied. The second phase is the conversion of data X into a 2D matrix using the NumPy method. The third phase is the initialization of weight for our proposed model, using the Gaussian variable and L_y represent the number of layers and for the number of features shown as n. Weight matrix represented as W_j and the Weight matrix dimension as mxn. In the fourth phase, a 2Dim matrix D_2 dataset is converted into 3Dim matrix D_3 using reshape matrix function RM, Then our data is ready to feed to the Convolution Neural network (CNN) model. Furthermore, CNN produces the feature-maps F_m. These feature-maps are now converted into several vectors V_s after the max-pooling layer. These V_s then pass to the GRU model as input. GRU model analyzed the relation of aspect and context $R(ac)$ between each data sequence. After

TABLE III
COMPARISON WITH BASELINE METHODS

Attack type	F1-score	Precision	Recall
DOS	94.06	93.89	94.22
Fuzzy Attack	95.09	94.88	95.31
Impersonation Attack	94.01	93.77	94.25
Attack Free	93.79	93.69	93.91

that this information is then used by the Attention layer to assign-scores S_w to each sequence vector S_v. Moreover, S_v vectors are multiplied with their respective S_w scores and summed up in the form of context vector C_xV. In the Last stage, C_xV is used to predict the target class TL which is our attack title.

V. RESULTS

The F1-score, Recall, and Precision of different baseline methods against *CANintelliIDS* are reported in Table III, IV and VI, where the best and optimum performance metrics values are highlighted in bold. The Gain variable reports the improvement of *CANintelliIDS* over the state-of-the-art baseline method. The input data shape for each baseline method and the proposed model is different. We transformed, preprocessed, and normalized the dataset according to each baseline method. We trained our *CANintelliIDS* model with single anomaly types of data sequences using each class and mixed anomaly types trained as normal data denoted as class 0 and multiple readings of a known intrusion denoted as class 1. The graphs 11a, 11b, 11c and 11d show the accuracy curves of *CANintelliIDS*. Graphs show the train and test the accuracy of the model. Furthermore, other metrics and statics are discussed in the upcoming sections.

A. Results Under Single Attack Data Sequence

To evaluate the *CANintelliIDS* model against the DOS attack, *CANintelliIDS* is trained on the targeted messages and can be distinguished using the identifier $0x000$. Considering the DOS attack detection performance of the CARGU model, the following is noted. The results illustrate that the CARGU model outperforms in detecting attacks. The general percentage of correct predictions for detecting the attack are discussed as accuracy. order to prove that the training process is stable and it will always converge, we ran the same experiments multiple times and reported the average case and best case. In Figure 11 a, the model achieves a test accuracy of 94.06% for the known intrusion while detecting the DoS attacks on the system. In Figure 11 b, the model achieves a test accuracy of 95.09% while detecting the Fuzzy attacks on the system. In Figure 11 c, the model achieves a test accuracy of 94.01% while detecting the Impersonation attacks on the system. In Figure 11 d, the model achieves a test accuracy of 93.79% while detecting the Free attacks-states on the system. This shows that our model has successfully detected single attacks using a time series sequence with an average F1-score of 94%.

TABLE IV
PERFORMANCE COMPARISON OF *CANINTELLIIDS* WITH CONVENTIONAL APPROACHES. THE OVERALL BEST PERFORMER IS SHOWN IN BOLD

Approaches	F1-score	Precision	Recall
Random Forest	81.17	80.79	81.55
Logistic Regression	79.02	78.36	79.71
Support Vector Machine	83.65	83.21	84.11
Deep Neural Network	81.57	81.13	82.02
CNN	84.58	84.77	85.01
CANintelliIDS	**93.79**	**93.69**	**93.91**
Gain	**9.21**	**8.92**	**8.9**

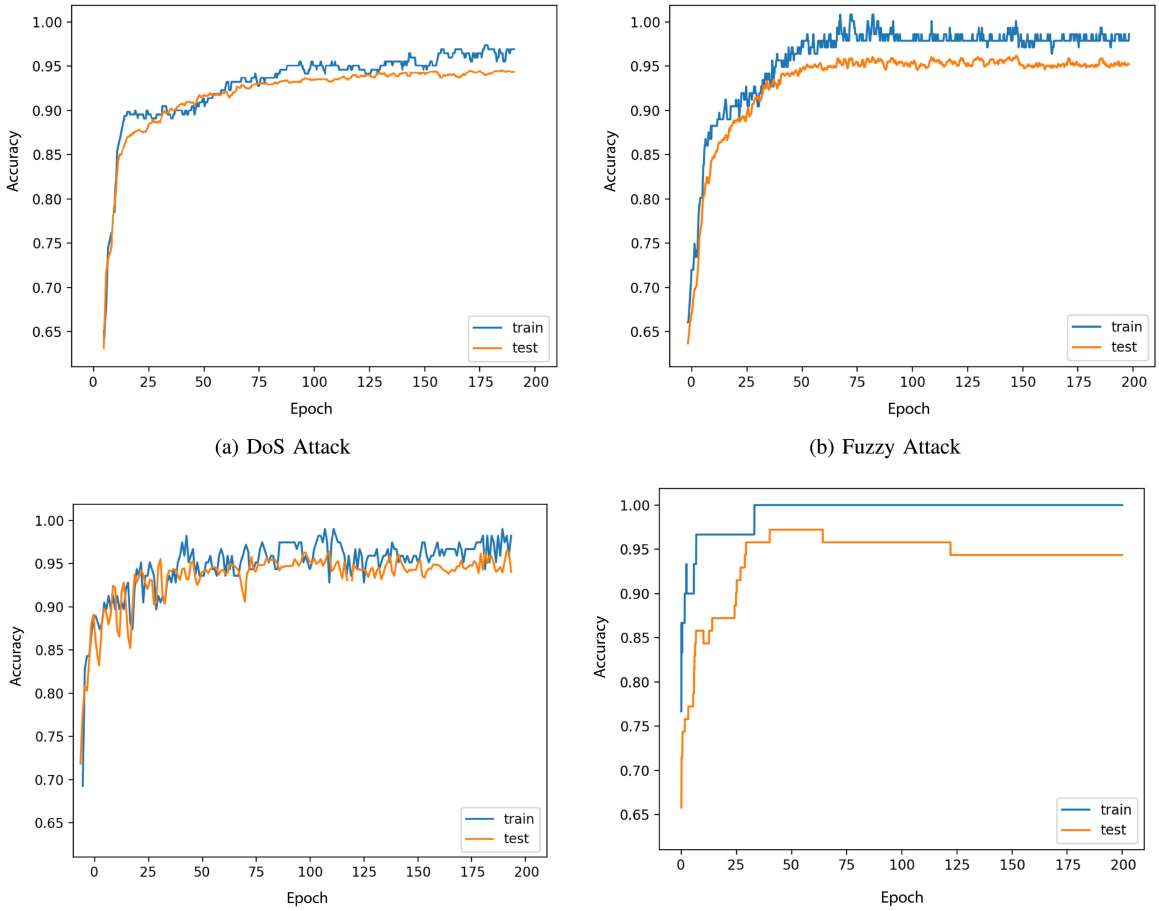
Table III shows the performance comparison of attacks achieved by *CANintelliIDS*. The evaluations are criteria is based on performance metric which includes F1-score, Precision, and Recall when each attack is detected using a time series sequence. DOS, fuzzy, impersonation attack, and attack free state are detected with the F1-score of 94.06, 94.8, 93.77, and 93.69 respectively.

B. Results Under Mixed Attack Data Sequence

In this section, the results of mixed sequence attacks are discussed. Table IV show the comparison of *CANintelliIDS* metric evaluation with other classification models. *CANintelliIDS* achieves the best proficiency counter to other assessed methods, resulting in an inclusive F1-score of 97.13% and 93.79% for training and testing intrusions respectively. The improvement of our approach over the best baseline approach (RNN+Heuristics) attained up to 35.01% in F1-score for test intrusions. A remarkable Recall (93.91%) and Precision (93.69%) score of *CANintelliIDS* show the capability of *CANintelliIDS* that it can competently identify most of the intrusions with a very less uncertainty rate of detecting false-positive.

Table V demonstrate the results of state-of-the-art *CANintelliIDS* in comparison with baseline models. These baseline models are compared based on F1-score, Precision, and Recall. The metrics mentioned for each entity are their capability while detecting test intrusions. OTIDS [6] is threshold-based method, OCSVM [35] and IF [36] are both classification based models, and RNN+Heuristics [10] and CANTransfer [19] are both Ensemble-based models. *CANintelliIDS* performs better than baseline approaches, it can also be observed that *CANintelliIDS* performs a step-ahead than the recent state-of-the-art approach [19] with a notable gain of 5.32%, 5.72% and 4.94% in F1-score, Precision, and Recall respectively.

Table VI shows detailed metric evaluation with [19]. The known Intrusion column is the test case when the intrusion is created and tested from the dataset and the test intrusion column represents the intrusions out of the dataset being executed at the proposed approach. Class 0 denotes normal reading while class 1 denotes the attack readings. *CANintelliIDS* achieves the best proficiency counter to the One-shot Learning model proposed by Tariq *et al.* [19], resulting in an inclusive F1-score of 97.13% and 93.79% for cross-validation intrusions. The improvement of our approach over the best baseline



(a) DoS Attack (b) Fuzzy Attack

(c) Impersonation Attack (d) Free Attack

Fig. 11. Result of 4 Attacks on 200 epochs.

TABLE V
PERFORMANCE COMPARISON OF *CANINTELLIIDS* WITH THE STATE-OF-THE-ART RECENT BASELINE METHODS. OVERALL BEST PERFORMER IS SHOWN AS BOLD

Approaches	F1-score	Precision	Recall
OCSVM[35]	16.55	10.83	35.12
IF[36]	20.90	15.62	31.56
OTIDS[6]	52.73	70.81	42.01
RNN+Heuristics[10]	58.78	70.25	50.53
CANTransfer[19]	88.47	87.97	88.97
DCNN[26]	88.47	87.97	88.97
CANintelliIDS	**93.79**	**93.69**	**93.91**
Gain	**5.32**	**5.72**	**4.94**

approach (CNN+AGRU) attained up to 11.79% in F1-score for test intrusions. A remarkable Recall (93.91%) and Precision (93.69%) score of *CANintelliIDS* show the capability of *CANintelliIDS* that it can competently identify most of the intrusions with a very less uncertainty rate of detecting false-positive.

VI. OBSERVATION AND DISCUSSION

The main objective of this research is to identify intrusion detection attacks. For this purpose, we utilized several deep learning and conventional machine learning approaches. Firstly, we tested machine learning techniques to achieve promising results. We apply logistic regression, random forest, and support vector machine and precisely tuned these models. We achieved the highest F1-score of 83% which is not promising. Then, we develop a Neural network consisting of 5 hidden layers but unfortunately get less accuracy than the Support vector machine due to the complex structure of data points. Therefore, we switched to Deep learning models. For this purpose, we converted our data points into vector sequences. At first, we built the CNN model for this classification task and got a better score this give us more clear motivation to further improved our results it required a more generalized model so to achieve the start of the are results we moved to the attention-based sequence model (AGRU) which perform very best but it only work with relational aspect and contextual information so extract the best features we embed the CNN layers without AGRU model and this way we achieve the state-of-the-art results as shown in this Table VI, In Known intrusion (Training Score), we achieve 98.21% F1-Score for Class 0 (benign) and 97.13% F1 score for Class 1 (attack). In the testing phase, we achieve a 94.38% F1 Score with Class 0 (benign) and a 93.79% F1 score for Class 1 (attack). Overall Gain in test intrusion attack detection, we achieve 4.38% for Class 0 (benign) and 5.78% for Class 1 (attack).

TABLE VI
RESULT OF TRAINING VS TESTING INTRUSION IN COMPARISON WITH [19]. KEY: CLASS 0 - NORMAL, CLASS 1 - ATTACK

		Known Intrusion			Test Intrusion		
	Class	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Tariq et al. [19] One-shot Learning	0	99.0%	91.0%	95.0%	92.0%	91.0%	91.0%
	1	90.0%	90.0%	94.0%	81.0%	83.0%	82.0%
CANintelliIDS	0	98.17%	98.26%	98.21%	94.18%	94.59%	94.38%
	1	96.91%	97.36%	97.13%	93.69%	93.91%	93.79%
Gain	0	-0.2%	7.26%	3.21%	2.18%	3.59%	2.79%
	1	6.91%	7.36%	3.13%	12.69%	10.91%	10.79%

VII. CONCLUSION

The objective of this paper is to develop an approach to detect intrusion attacks on the CAN bus to improve their reliability. Our outcomes show that the use of deep learning models, for example, CNN to recognize such attacks (i.e., DOS, Fuzzy, and Impersonation assaults) in CAN bus is a practical way and it can improve upon the well-established techniques such as GRU with an attention mechanism. Moreover, our outcomes show that a combination of CNN and GRU increases the detection performance of attack detection and subsequently brings about improved detection performance. More precisely, we show that by using a CNN empowered attention based GRU on CAN bus CAN data sequences it is possible to detect the attack with high Accuracy, Recall, Precision, and F1 score. We believe that the proposed detection approach contributes to improving vehicle security without bringing any change to the CAN protocol.

We intend to work on making the *CANintelliIDS* even more advanced by making it intelligent enough such that it may detect attack to mitigate the damage at the earlier stages. We also intend to work on detecting anomalous and faulty sensor nodes. Least but not least, the next version of this approach will be capable of covering the deficiency in the current performance of the model. Ultimately, We intend to make a dataset having DOS, Fuzzy, and Impersonation attacks which would be challenging enough to help out researchers in making CAN defense mechanisms more proficiently to encounter the latest cyber-attacks.

REFERENCES

[1] K. Pazul,"Controller area network (can) basics," *Microchip Technol. Inc*, vol. 1, pp. 1–9, 1999.

[2] "Controller area network (Can Bus)," 2020, Accessed: May 20, 2020. [Online]. Available: https://en.wikipedia.org/wiki/CAN_bus

[3] N. Navet and F. Simonot-Lion, "A review of embedded automotive protocols," *Automotive Embedded Syst. Handbook*, Ind. Inf. Technol. Ser., pp. 4–1, 2008.

[4] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Commun. Surv. Tut.*, vol. 15, no. 2, pp. 860–880, Second Quarter, 2012.

[5] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Proc. IEEE Int. Conf. Inf. Net*., 2016, pp. 63–68.

[6] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust*, 2017, pp. 57–5709.

[7] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (can) bus system: A review," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, 2019, Art. no. 184.

[8] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Trans. Intelli. Transport. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[9] T. Hoppe, S. Kiltz, and J. Dittmann, "Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats," in *Proc. Int. Conf. Comput. Safety, Rel., Secur.*, 2009, pp. 145–158.

[10] S. Tariq, S. Lee, H. K. Kim, and S. S. Woo, "Detecting in-vehicle can message attacks using heuristics and rnns," in *Proc. Int. Workshop Inf. Oper. Technol. Secur. Syst.*, 2018, pp. 39–45.

[11] A. Boudguiga, W. Klaudel, A. Boulanger, and P. Chiron, "A simple intrusion detection method for controller area network," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.

[12] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Des. Test*, vol. 36, no. 6, pp. 48–55, Dec. 2019.

[13] G. Dupont, J. Den Hartog, S. Etalle, and A. Lekidis, "Evaluation framework for network intrusion detection systems for in-vehicle can," in *Proc. IEEE Int. Conf. Connected Veh. Expo.*, 2019, pp. 1–6.

[14] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ecus using inimitable characteristics of signals in controller area networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 4757–4770, Jun. 2018.

[15] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, "Deepids: Deep learning approach for intrusion detection in software defined networking," *Electron.*, vol. 9, no. 9, 2020, Art. no. 1533.

[16] N. Guo, Y. Tian, F. Li, and H. Yang, "Attention-based deep learning for network intrusion detection," in *Proc. Int. Conf. Image, Video Process. Artif. Intelli.*, 2020, p. 115841I.

[17] W. Wu *et al.*, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.

[18] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal*., 2016, pp. 130–139.

[19] 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. 35th Annu. ACM Symp. Appl. Comput.*, 2020, pp. 1048–1055.

[20] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr, and S. J. Prowell, "Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: A data-driven approach to in-vehicle intrusion detection," in *Proc. 12th Annu. Conf. Cyber Inf. Secur. Res.*, 2017, pp. 1–4.

[21] R. Buttigieg, M. Farrugia, and C. Meli, "Security issues in controller area networks in automobiles," in *Proc. 18th Int. Conf. Sci. Techn. Autom. Control Comput. Eng.,* 2017, pp. 93–98.

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

[23] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Netw.*, vol. 31, no. 5, pp. 50–58, 2017.

[24] H. Olufowobi, G. Bloom, C. Young, and J. Zambreno, "Work-in-progress: Real-time modeling for intrusion detection in automotive controller area network," in *Proc. IEEE Real-Time Syst. Symp.*, 2018, pp. 161–164.

[25] Z. Niu, Z. Yu, W. Tang, Q. Wu, and M. Reformat, "Wind power forecasting using attention-based gated recurrent unit network,"*Energy*, vol. 196, 2020, Art. no. 117081.

[26] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, 2020, Art. no. 100198.
[27] M. Tang, M. Alazab, and Y. Luo, "Big data for cybersecurity: Vulnerability disclosure trends and dependencies," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 317–329, Sep. 2019.
[28] S. Venkatraman and M. Alazab, "Use of data visualisation for zero-day malware detection," *Secur. Commun. Netw.*, 2018.
[29] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, and Z. Qin, "Mthael: Cross-architecture iot malware detection based on neural network advanced ensemble learning," *IEEE Trans. Comput.*, vol. 69, no. 11, pp. 1654–1667, Nov. 2020.
[30] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, 2020, Art. no. 107138.
[31] M. Abadi, *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th {USENIX} Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.
[32] A. H. Nguyen, H. T. T. Tran, T. C. Thang, and Y. M. Ro, "Fast recognition of human actions using autocorrelation sequence," in *Proc. IEEE 7th Glob. Conf. Consum. Electron.*, 2018, pp. 114–115.
[33] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," 2017, *arXiv:1704.02971*.
[34] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 577–585.
[35] L. M. Manevitz and M. Yousef, "One-class SVMS for document classification," *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, Dec. 2001.
[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.