

# CAN-BERT do it? Controller Area Network Intrusion Detection System based on BERT Language Model

Natasha Alkhatib, Maria Mushtaq, Hadi Ghauch, Jean-Luc Danger  
Télécom Paris, IP Paris, Palaiseau, France

{natasha.alkhatib, maria.mushtaq, hadi.ghauch, jean-luc.danger}@telecom-paris.fr

**Abstract**—Due to the rising number of sophisticated customer functionalities, electronic control units (ECUs) are increasingly integrated into modern automotive systems. However, the high connectivity between the in-vehicle and the external networks paves the way for hackers who could exploit in-vehicle network protocols’ vulnerabilities. Among these protocols, the Controller Area Network (CAN), known as the most widely used in-vehicle networking technology, lacks encryption and authentication mechanisms, making the communications delivered by distributed ECUs insecure. Inspired by the outstanding performance of bidirectional encoder representations from transformers (BERT) for improving many natural language processing tasks, we propose in this paper “CAN-BERT”, a deep learning based network intrusion detection system, to detect cyber attacks on CAN bus protocol. We show that the BERT model can learn the sequence of arbitration identifiers (IDs) in the CAN bus for anomaly detection using the “masked language model” unsupervised training objective. The experimental results on the “Car Hacking: Attack & Defense Challenge 2020” dataset show that “CAN-BERT” outperforms state-of-the-art approaches. In addition to being able to identify in-vehicle intrusions in real-time within 0.8 ms to 3 ms w.r.t CAN ID sequence length, it can also detect a wide variety of cyberattacks with an F1-score of between 0.81 and 0.99.

**Index Terms**—controller area network, CAN, Intrusion Detection, bidirectional encoder representations from transformers, BERT, in-vehicle network, cyberattacks.

## I. INTRODUCTION

To fulfill automotive features, the Controller Area Network (CAN) bus is widely used as the de-facto standard for message communication between different electronic control units (ECUs) in today’s vehicles. It is sometimes referred to as a “message-based” system, since it focuses on the transmission of diagnostic, informative and controlling data through messages, also known as CAN data frames. In fact, while developing a vehicle, all conceivable CAN bus messages and their respective priority, encoded into an identifier called “CAN ID”, must be determined beforehand. Due to the lack of authentication, any device can connect physically or wirelessly to the CAN bus, broadcast CAN data frames, and all the participants on the CAN bus can receive it without verifying its source. Consequently, since CAN security was not a design priority, many message injection attacks have become widely implemented. These attacks can interfere with the desired function of the system, shut down some connected nodes,

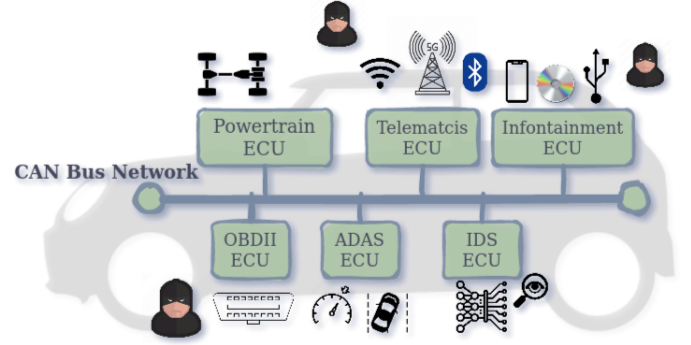


Fig. 1. A CAN bus network exploited by attackers through different physical and wireless interfaces.

and make the vehicle behave abnormally, putting at risk the safety of the driver and the passengers. To address these security flaws, researchers have proposed intrusion detection as a supplementary layer of protection to specialized security solutions. By monitoring the communication between different ECUs within a CAN bus system, a network intrusion detection system (N-IDS) can detect deviations from the normal message exchange behavior and, thereby, identify both anticipated and novel cyberattacks. The adoption of deep neural networks for in-vehicle intrusion detection have lately proliferated, with impressive results. Since a message injection attack can alter the normal order of occurrence of several CAN IDs, researchers have deployed deep learning based sequential models, to model the CAN ID sequences. Some studies have proposed the use of autoregressive models that are trained to capture the patterns of regular CAN ID sequences by predicting the future CAN ID sequence based on the preceding one, such as recurrent neural network (RNN) models and its variants and the generative pretrained transformer (GPT). However, these models identify malicious network intrusions on CAN ID traffic by focusing primarily on the exchange of CAN ID messages from earlier steps rather than integrating the left and right context of a CAN ID sequence, limiting the model’s capacity to grasp the whole context information representation. Additionally, these algorithms focus largely on the correlation

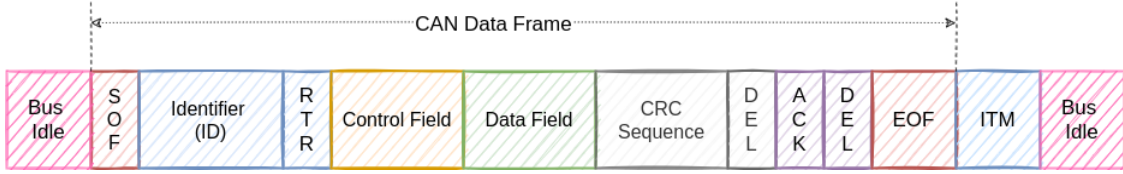


Fig. 2. CAN packet structure.

between CAN ID messages in normal sequences, which would result in false alarms for intrusion detection whenever the correlation is breached. Hence, due to these limitations, the autoregressive models do not adequately depict the natural communication behavior between the various ECUs.

To address these challenges, we propose CAN-BERT, an intrusion detection system based on a language representation model called Bidirectional Encoder Representations from Transformers (BERT). CAN-BERT, in contrast to autoregressive models, is a self-supervised model which can successfully depict deep bidirectional representations from CAN ID sequences by conditioning on both left and right context in its various layers. By using the “masked language model” unsupervised training objective, CAN-BERT model masks some of the CAN IDs in the input at random, with the goal of predicting the conventional ID of the masked word based on its left and right context.

We evaluate our approach using the recently published “Car Hacking: Attack & Defense Challenge 2020” collected from three different cars, Chevrolet Spark, Hyundai Sonata and Kia Soul and which contains diverse types of message injection attacks.

Our contributions are summarized below:

- Inspired by the outstanding performance of BERT model for improving many natural language processing tasks, we propose “CAN-BERT”, a novel BERT-based intrusion detection system architecture which can detect known and unprecedented cyberattacks in CAN ID sequences.
- We evaluate the performance of our approach with the recently published “Car Hacking: Attack & Defense Challenge 2020” collected from three different cars, Chevrolet Spark, Hyundai Sonata and KIA Soul and which contains diverse types of message injection attacks. We also compare our model with other baseline models.

Towards this end, our paper is organized into six sections. In Section III, we present an overview of the Controller Area Network (CAN) and the Bidirectional Encoder Representations from Transformers model (BERT). In Section IV, we present an overview of our proposed framework “CAN-BERT”. Section V discusses the launched experiments with the corresponding dataset and the proposed metrics for IDS’ evaluation. In Section VI, we discuss the obtained results showing the proposed model accuracy and complexity. Finally, we conclude our paper with future work direction.

## II. RELATED WORK

Intrusion Detection systems (IDSs) have been widely used to detect intrusions on the Controller Area Network (CAN). Intrusions can be detected either by inspecting the content or the signals transmitted by CAN data frames [19], or by examining the order by which different CAN data frames’ identifiers are exchanged between the ECUs [20].

## III. PRELIMINARIES

### A. Controller Area Network

The Controller Area Network (CAN), created by BOSCH in 1983, is a potent networking technology essential for the development of useful automotive features. Due to its robustness represented by its ability to allow various ECUs to be connected in almost all areas of a car, it still prevails in vehicles today. As seen in Figure, it is a bus system, meaning that all Electronic Control Units (ECUs) share the same wiring.

It is a “message-based” system in which messages, also known as CAN data frames, are transmitted between various ECUs. As depicted in Figure. 2, each CAN data frame is composed of the following elements: Start of frame (SOF), identifier (ID), Remote frame transmission field (RTR), control field, data, cyclic redundancy check (CRC), delimiters (DEL) and acknowledgment fields (ACK), and end of frame fields. Each CAN bus message has a priority that is represented by the arbitration identifier field (ID) that can either be composed of 11 or 29 bits, depending on the car manufacturer and which will be mainly used in our work for detecting in-vehicle intrusions.

To avoid contention between multiple ECUs willing to transmit CAN messages in the medium, CAN employs a priority-based mechanism which allows the ECU with the highest priority/lowest value identifier to transmit before others. The procedure is termed “arbitration” because the message with the highest priority wins out over competing messages with a lower priority at the time of transmission.

1) *Security Weaknesses*: CAN does not prohibit several ECUs from sharing the same IDs. Moreover, CAN messages are broadcast and do not contain any sender’s address. Consequently, any device linked to the CAN bus can use any pre-defined ID, communicate its message without authentication or encryption, and all associated ECUs can receive it. The receiver defines whether or not a message identification causes the receiving ECU to retain and process the given data. Consequently, an attacker is able to broadcast spoofed CAN messages, eavesdrop on all the CAN traffic and collect detailed

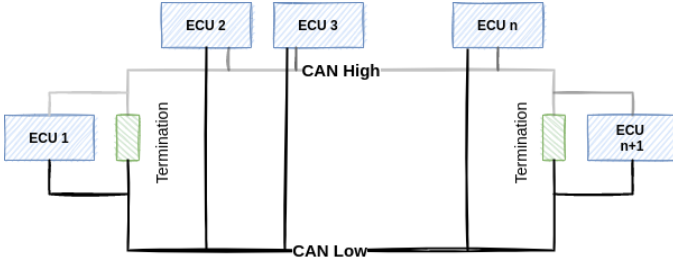


Fig. 3. ECU devices connected through CAN bus, source [1].

information about it, resulting in Fuzzing and Malfunction attacks.

Additionally, as previously mentioned, the CAN bus leverages the arbitration method which discerns between "dominant" (0) and "recessive" (1) bits in the message identifiers. Therefore, if several ECUs begin transmitting simultaneously, the ECU whose message begins with a greater number of dominant "0" bits will take over the CAN bus. As soon as a unit detects that the message on the bus is no longer the message it is transmitting, it halts its transmission, waits for the real transmission to conclude, then waits for the interframe gap to expire and retransmits its message. This phenomenon carries the risk that a message with a lower priority will never be delivered if the network is very congested and can be exploited by attackers to launch denial of service (DoS)/flooding attacks.

## B. BERT

Bidirectional Encoder Representations from Transformers (BERT), proposed by Devlin et al. [2], is a state-of-the-art language representation model which is designed to pretrain bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Regarding its architecture, it is a multi-layer bidirectional Transformer encoder based on the original implementation proposed by Vaswani et al. [3].

Inspired by its outstanding performance in modeling sequential data, BERT is recently employed for sequence anomaly detection [6] [7] [8] [9] [10]. To the best of our knowledge, none of the previous works have evaluated the performance of the BERT model for in-vehicle intrusion detection on CAN protocol.

In order to detect anomalies in sequences, it's crucial to incorporate context from both left and right directions of the sequence. Sequential anomalies may be misdetected by traditional unidirectional models, such as OpenAI GPT and RNNs, where every token can only attend to context to its left. To solve this significant constraint, some researchers have proposed a shallow concatenation of both left-to-right and right-to-left architecture of the autoregressive models, such as Bi-RNN and Bi-GPT [5]. However, these approaches aren't as powerful as BERT which adopts a "masked language model" (MLM) training objective, in which input sequence tokens are randomly masked and the goal is to predict the

original vocabulary id of the masked word based on its context. In contrast to denoising auto-encoders, BERT predicts the masked words instead of reconstructing the whole sequence [12].

## IV. PROPOSED FRAMEWORK: CAN-BERT

We propose "CAN-BERT", a pattern-based anomaly detection algorithm, which leverages a BERT-based architecture to detect message injection intrusions in the CAN bus. As seen in Figure. 4, our model is composed of a multi-layer bidirectional Transformer encoder and is trained using the "masked language model" self-supervised task to model normal CAN ID sequences. The following subsections elaborately describe the suggested framework.

### A. Model description

Note that  $\mathbf{S} = [\mathbf{id}_1, \dots, \mathbf{id}_t, \dots, \mathbf{id}_T]$  is an observed sequence of  $T$  CAN identifiers, arranged in their order of transmission in the CAN bus network, where an identifier  $\mathbf{id}_t \in \mathbb{ID}$  is an  $M$ -dimensional vector which denotes the CAN ID transmitted at time  $t$  by an ECU,  $\mathbb{ID}$  indicates the set of CAN IDs extracted from CAN messages, and  $M$  is the size of the  $\mathbb{ID}$  set.

Since anomaly detection is an unsupervised learning-based technique in which only normal data are used for training, a collection of  $N$  CAN ID sequences, represented as  $\mathbb{D}_{training} = \{\mathbf{S}_1, \dots, \mathbf{S}_j, \dots, \mathbf{S}_N\}$ , is solely used as a training dataset.

**Identifier Embeddings** To feed the appropriate input to the BERT model, each identifier  $\mathbf{id}_t^j$  with size  $(M, 1)$  in a CAN sequence  $\mathbf{S}^j$  is firstly projected into a  $d$ -dimensional space using a single linear layer, i.e.:

$$\mathbf{e}_t^j = \mathbf{W}^e \mathbf{id}_t^j + \mathbf{b}^e, \forall i \in \{1..T\}, \forall j \in \{1..N\} \quad (1)$$

where  $\mathbf{e}_t^j$  represents the identifier embedding with size  $(d, 1)$ ,  $\mathbf{W}^e \in \mathbb{R}^{d \times M}$  is the input embedding weight matrix, and  $\mathbf{b}^e \in \mathbb{R}^d$  denotes the bias. Both  $\mathbf{W}^e$  and  $\mathbf{b}^e$  are trainable parameters.

Subsequently, the identifier's position is encoded into a  $d$ -dimensional positional embedding  $\mathbf{p}_t^j$  using a sinusoidal function. To this end, the CAN ID, fed into the CAN-BERT model, is a summation of both the positional encoding and the input embedding :

$$\mathbf{x}_t^j = \mathbf{e}_t^j + \mathbf{p}_t^j \quad (2)$$

where  $\mathbf{x}_t^j$  is the total embedding  $j$ -th identifier in the  $t$ -th CAN ID sequence  $\mathbf{id}_t^j$ , thereby the convergence of the input sequence  $\mathbf{S}^j$  into  $\mathbf{X}^j = [\mathbf{x}_1^j, \dots, \mathbf{x}_T^j]^T$  with  $\mathbf{X}^j$  a matrix with size  $T \times d$ .

**Transformer Encoder** The encoded input  $\mathbf{X}^j$  is then delivered into a stack of  $L$  transformer encoder layers, each of which has two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network [3]. A residual connection is employed around each of these two sub-layers, followed by layer normalization [4], as follows:

$$\begin{aligned} \mathbf{E}^{(j,l)} &= g(\mathbf{X}^{(j,l)}) + f(\mathbf{X}^{(j,l)} + g(\mathbf{X}^{(j,l)})) \\ \mathbf{H}^{(j,l)} &= z(\mathbf{E}^{(j,l)}) + f(\mathbf{E}^{(j,l)} + z(\mathbf{E}^{(j,l)})) \\ \mathbf{X}^{(j,l+1)} &= \mathbf{H}^{(j,l)}, \forall l < L \end{aligned} \quad (3)$$



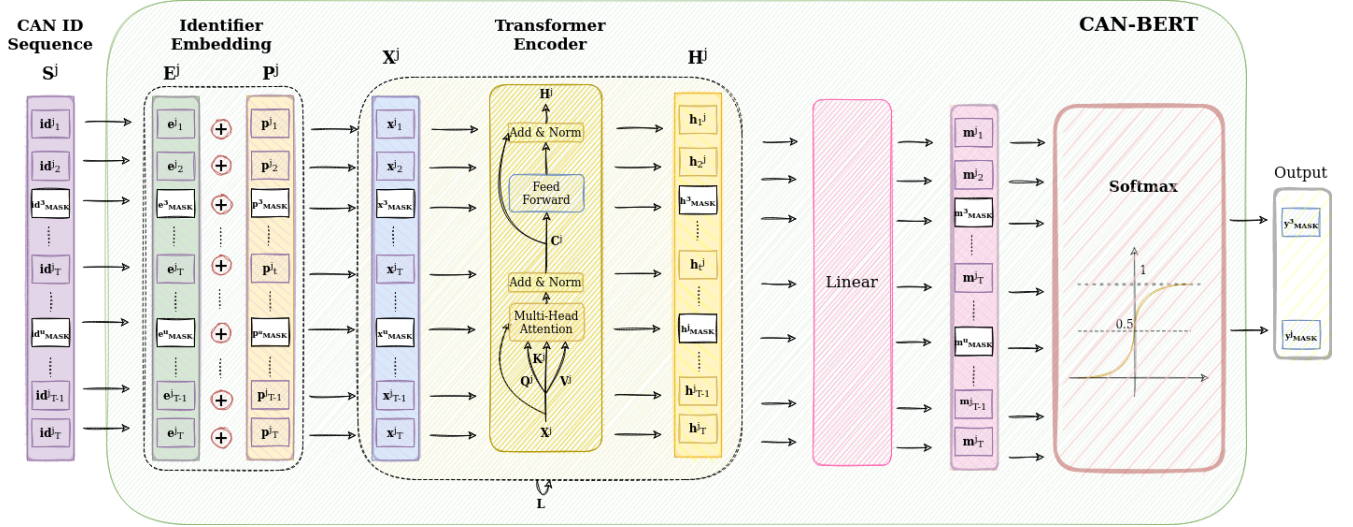


Fig. 4. CAN-BERT model architecture

where  $\mathbf{E}^{(j,l)}$  represents the output of the first sublayer for the  $l$ -th transformer encoder layer with size  $T \times d$ ,  $\mathbf{H}^{(j,l)}$  represents the output of the second sublayer for the  $l$ -th transformer encoder layer with size  $T \times d$ ,  $g$  is the multi-head attention function,  $z$  is the position wise feed forward function, and  $f$  is the layer normalization function.

**Attention** We use the scaled dot-product attention proposed by [3], requiring query  $\mathbf{Q}^{(j,l)}$ , key  $\mathbf{K}^{(j,l)}$ , and value  $\mathbf{V}^{(j,l)}$  representations, and which are projections of the embedded sequence  $\mathbf{X}^{(j,l)} \in \mathbb{R}^{T \times d}$ . In fact, we leverage the dot-product similarity to compare the query representation of a given CAN identifier to all other keys. Hence, if the query and key are comparable have a high attention weight, the matching value is deemed to be relevant. The output is therefore computed as a weighted sum of the values  $\mathbf{V}$ :

$$\text{Attn}(\mathbf{Q}^{(j,l)}, \mathbf{K}^{(j,l)}, \mathbf{V}^{(j,l)}) = \sigma\left(\frac{\mathbf{Q}^{(j,l)} \mathbf{K}^{(j,l)T}}{\sqrt{d}}\right) \mathbf{V}^{(j,l)} \quad (4)$$

$$\text{Attn}(\mathbf{Q}^{(j,l)}, \mathbf{K}^{(j,l)}, \mathbf{V}^{(j,l)}) = \mathbf{A} \mathbf{V}^{(j,l)}$$

where  $\sigma$  is the softmax function,  $\mathbf{A} \in \mathbb{R}^{T \times T}$  denotes the attention weight matrix containing attention weights, and  $d$  is the dimension of the  $\mathbf{Q}^{(j,l)}$ ,  $\mathbf{K}^{(j,l)}$ ,  $\mathbf{V}^{(j,l)}$  vectors.

As described by [3], the multiple heads of attention allows the model to concurrently capture diverse aspect of data at distinct CAN IDs. Hence, we adopt a multi-head attention (MHA) mechanism in which the  $d$ -dimensional CAN identifiers are projected into subspaces calculated by different attention heads  $n \in \{1, \dots, H\}$ :

$$\begin{aligned} \mathbf{Q}^{(j,n,l)} &= \mathbf{X}^{(j,l)} \mathbf{W}^{(Q,n)}, \mathbf{Q}^{(j,n,l)} \in \mathbb{R}^{T \times F} \\ \mathbf{K}^{(j,n,l)} &= \mathbf{X}^{(j,l)} \mathbf{W}^{(K,n)}, \mathbf{K}^{(j,n,l)} \in \mathbb{R}^{T \times F} \\ \mathbf{V}^{(j,n,l)} &= \mathbf{X}^{(j,l)} \mathbf{W}^{(V,n)}, \mathbf{V}^{(j,n,l)} \in \mathbb{R}^{T \times F} \end{aligned} \quad (5)$$

where  $\mathbf{Q}^{(j,n,l)}$ ,  $\mathbf{K}^{(j,n,l)}$  and  $\mathbf{V}^{(j,n,l)}$  are the query, key and value vectors, respectively of the  $j$ -th CAN ID sequence for

the  $l$ -th transformer encoder layer and which are calculated using the  $n$ -th attention head. The  $\mathbf{W}^{(Q,n)}$ ,  $\mathbf{W}^{(K,n)}$  and  $\mathbf{W}^{(V,n)}$  are their corresponding trainable weight matrices  $\in \mathbb{R}^{d \times F}$ , and  $F$  is set to  $D/H$ . The results are then concatenated and projected back into representation space using the weight matrix  $\mathbf{W}^o \in \mathbb{R}^{HF \times D}$  as follows:

$$\text{head}_n^{(j,l)} = \text{Attn}(\mathbf{Q}^{(j,n,l)}, \mathbf{K}^{(j,n,l)}, \mathbf{V}^{(j,n,l)}) \quad (6)$$

$$\bar{\mathbf{X}}^{(j,l)} = [\text{head}_1^{(j,l)}, \dots, \text{head}_n^{(j,l)}, \dots, \text{head}_H^{(j,l)}] \mathbf{W}^o \quad (7)$$

where  $\bar{\mathbf{X}}^{(j,l)} \in \mathbb{R}^{(T,d)}$ .

**Position-wise feed-forward** A position-wise feed-forward network with a ReLU activation is thereby applied to each representation in each of the layers of our encoder, in addition to attention sub-layers, using the following equation:

$$z(\mathbf{E}^{(j,l)}) = [\mathbf{W}_1 \mathbf{E}^{(j,l)}]_+ \circ \mathbf{W}_2 \quad (8)$$

where  $\mathbf{E}^{(j,l)}$  is previously defined in (3),  $\mathbf{W}_1$  and  $\mathbf{W}_2$  are trainable projection matrices, where  $\circ$  is the hadarmard product, and  $[\ ]_+$  is the element-wise maximum.

After passing through different transformer layers, the  $L$ -th contextual embedding vectors of the CAN IDs, denoted as  $\mathbf{h}_t^{(j,L)}$  with size  $(d, 1) \in \mathbf{H}^{(j,L)} = [\mathbf{h}_1^{(j,L)}, \dots, \mathbf{h}_T^{(j,L)}]^T$ , are fed into a single linear layer which projects them back to the  $M$ -dimensional layer, as follows:

$$\mathbf{m}_t^j = \mathbf{W}^m \mathbf{h}_t^{(j,L)} + \mathbf{b}^m, \forall i \in \{1..T\}, \forall j \in \{1..N\} \quad (9)$$

where  $\mathbf{m}_t^j$  represents the projected output with size  $(M, 1)$ ,  $\mathbf{W}^m \in \mathbb{R}^{M \times d}$  is the input embedding weight matrix, and  $\mathbf{b}^e \in \mathbb{R}^M$  denotes the bias. Both  $\mathbf{W}^m$  and  $\mathbf{b}^m$  are trainable parameters.

## B. Training and Inference

We use the masked language model training method to train the CAN-BERT model on capturing the patterns of normal CAN ID sequences. Hence, CAN sequences with random

mask as inputs, where we randomly replace a ratio of CAN IDs in a sequence with a specific MASK token, are fed into CAN-BERT. The purpose of training is to reliably anticipate the CAN IDs that have been randomly masked.

To achieve that, we feed the contextual embedding vector of the  $u$ -th MASK in the  $j$ -th CAN ID sequence  $\mathbf{m}_{[MASK_u]}^j$  to a softmax function, which will return a probability distribution for the whole set of CAN IDs  $\mathbb{ID}$ :

$$\hat{\mathbf{y}}_{[MASK_u]}^j = \sigma(\mathbf{W}^c \mathbf{m}_{[MASK_u]}^j + \mathbf{b}^c) \quad (10)$$

where  $\hat{\mathbf{y}}_{[MASK_u]}^j$  is an  $m$ -dimensional vector,  $\sigma$  is the softmax function,  $\mathbf{m}_{[MASK_u]}^j$  and  $\mathbf{b}^c$  are trainable parameters.

CAN-BERT is trained to minimize the cross entropy loss over a batch of  $I$  sequences (with  $I \leq N$ ), for masked CAN ID prediction, which is defined as:

$$\mathcal{L}_{MASK} = -\frac{1}{IR} \sum_{j=1}^N \sum_{u=1}^R \mathbf{y}_{[MASK_u]}^j \log \hat{\mathbf{y}}_{[MASK_u]}^j \quad (11)$$

where the ground-truth  $u$ -th CAN ID in the  $j$ -th sequence is denoted as  $\mathbf{y}_{[MASK_u]}^j$ ,  $R$  is the total number of masked tokens in the  $j$ -th sequence, and  $N$  is the number of training samples.

By modeling the normal exchange of messages through CAN bus using CAN-BERT, our model is expected, after training, to predict a candidate set with the normal CAN IDs having the highest likelihood for each masked token. Hence, for a randomly masked testing sequence, we calculate the probability distribution represented in (10), for each masked CAN ID. Therefore, if the actual CAN ID is among the anticipated candidates, the corresponding CAN ID sequence is considered as normal. Otherwise, it is deemed abnormal.

## V. EXPERIMENTAL SETTINGS

### A. Dataset

To assess the proposed CAN-BERT, we leverage the ‘‘In-Vehicle Network Intrusion Detection Challenge’’ dataset [13] (presented in Table I), which was used at the ‘‘In-vehicle Network Intrusion Detection track’’ of ‘‘Information Security R&D Data Challenge 2019’’. It includes normal and abnormal in-vehicle network traffic data of HYUNDAI Sonata, KIA Soul, and CHEVROLET Spark vehicles collected during their stationary state. We have mainly used its preliminary dataset, which includes three types of attacks (Flooding, Fuzzy, and Malfunction). The dataset is labeled and each sample is represented by the following features: ‘‘Timestamp’’ representing the logging time, ‘‘CAN ID’’ representing the CAN Identifier, ‘‘DLC’’ indicating the Data length code, and the Payload indicating the ‘‘CAN data’’ field.

1) *Attacks*: The dataset contains the following attacks:

- **Flooding Attack** The flooding attack was carried out by injecting many messages with the CAN ID set to 0x000 into the CAN network. Consequently, an ECU that transmits CAN data frames with such CAN ID dominates the CAN bus, which could restrict the communications among the ECU nodes and impair normal in-vehicle functions.

- **Fuzzy Attack** To implement fuzzy attacks, the attacker injected every 0.0003 seconds random CAN packets, for both the ID field and the Data field. This process lead to abnormal automotive functionalities behavior such as short beeping sound repeatedly occurring, the heater turning on, etc.
- **Malfunction Attack** The malfunction attack was carried out by injecting messages with a specified CAN ID from among the extractable CAN IDs of a particular vehicle in order to disable relevant automotive functions, such as IDs 0x316, 0x153 and 0x18E for the HYUNDAI YF Sonata, KIA Soul, and CHEVROLET Spark vehicles, respectively.

TABLE I  
IN-VEHICLE NETWORK INTRUSION DETECTION DATASET

Vehicle	Dataset	# Normal packets	# Abnormal packets	Size (MB)
CHEVROLET Spark	Attack Free	136,933	N/A	6.2
	Flooding	70,001	14,999	4.2
	Fuzzy	37,957	3,043	2.0
	Malfunction	47,005	3,995	2.5
HYUNDAI Sonata	Attack Free	117,172	N/A	5.8
	Flooding	78,907	17,093	4.9
	Fuzzy	78,905	9,095	4.5
	Malfunction	78,798	8,202	4.5
KIA Soul	Attack Free	192,515	N/A	9.3
	Flooding	103,928	16,072	6.2
	Fuzzy	122,387	21,613	7.4
	Malfunction	108,230	4,770	5.8

As mentioned in Section IV, we aim to detect if a sequence of ordered CAN ID contains injected messages. Hence, in order to represent CAN ID sequences, we use the **Feature-based Sliding Window (FSW)** to group CAN IDs which belong to the dataset into subsequences with fixed window size  $T$ , where  $T \in \{16, 32, 64, 128, 256\}$  and the slide size is 1. Moreover, each CAN ID sequence  $\mathbf{S} = [\mathbf{id}_1, \dots, \mathbf{id}_t, \dots, \mathbf{id}_T]$  has its corresponding labels represented by  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_t, \dots, \mathbf{y}_T]$  wherein each identifier  $\mathbf{id}_t \in \mathbf{S}$  is labeled as  $\mathbf{y}_t = 1$  if  $\mathbf{id}_t$  is an injected identifier in  $\mathbf{S}$  or as  $\mathbf{y}_t = 0$  otherwise. However, to identify the state of each sequence, we have used the following criteria:

$$z = \begin{cases} 1 \text{ (abnormal)} & \text{if } \exists \mathbf{y}_t = 1, \forall t \in \{1, \dots, T\} \\ 0 \text{ (normal)} & \text{otherwise} \end{cases}$$

where  $z$  is the CAN ID sequence’s label.

### B. Benchmark Models

The benchmark models for evaluating the performance of different CAN ID sequence anomaly detection algorithms with CAN-BERT on the chosen dataset, are detailed in this section.

- **Principal Component Analysis (PCA)**: PCA [14] is a feature selection model which can be used to reduce data features from  $m$  dimensions to  $n$ . Inverting the PCA transform does not retrieve the data lost during the application of the transform. The essence of PCA-based anomaly identification is that an anomalous sample

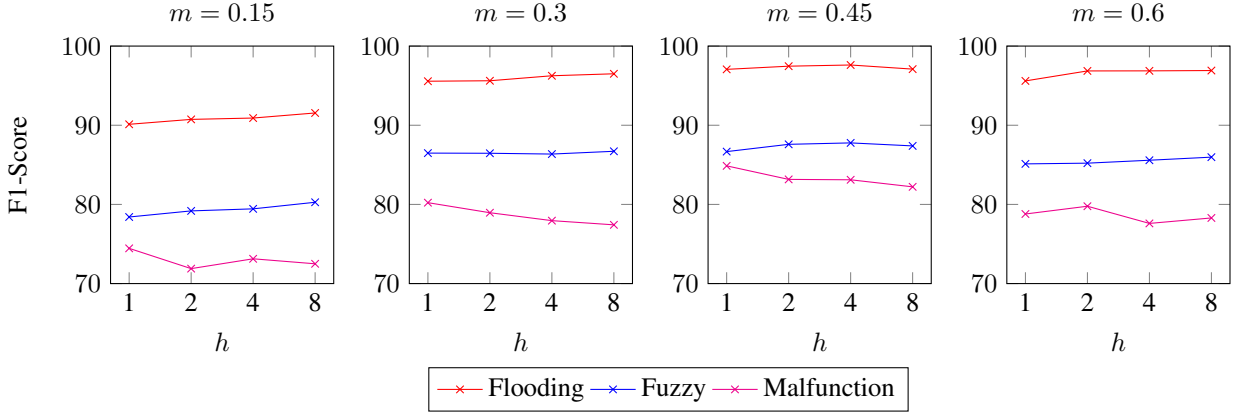


Fig. 5. Hyperparameter tuning the mask ratio  $m$  and the number of attention heads  $h$  on the “CHEVROLET Spark” dataset for  $T=32$ . We have obtained similar behavior pattern for the results w.r.t other sequence length and car types.

should have more loss or reconstruction error than a normal sample. In other words, the loss sustained when an anomalous sample is processed by a PCA algorithm and projected back to its dimension using PCA also should be greater than when the same procedure is performed on a normal sample.

- **Isolation Forest (iForest):** Isolation forest (IF), proposed by Liu et al. [15], detects anomalies using isolation rather than modelling the normal points. In fact, this technique presents a novel approach for isolating anomalies using binary trees, providing a new prospect for a speedier anomaly detector that directly targets abnormalities rather than profiling all regular instances.
- **Autoencoder (AE):** The autoencoder, introduced by Rumelhart et al. [16], is a deep learning based algorithm which seeks to learn a low-dimensional feature representation space suitable for reconstructing the provided data instances. During the encoding process, the encoder maps the original data onto low-dimensional feature space, while the decoder tries to retrieve the original data from the projected low-dimensional space. Reconstruction loss functions are used to learn the parameters of the encoder and decoder networks. Its reconstruction error value must be minimized during the training of normal instances and therefore used during testing as an anomaly score. In other words, compared to the typical data reconstruction error, anomalies that differ from the majority of the data have a large data reconstruction error. In our experiments, we have tested Long short-term based memory (LSTM) and Bidirectional LSTM (BiLSTM) models with different network hyperparameters: BiLSTM-AE-4 (with 4 layers), LSTM-AE-4 (with 4 layers), and LSTM-AE-8 (with 8 layers).

### C. Evaluation metrics

For measuring the performance of different anomaly-based IDS, we use the F1-score metric, a weighted average result of both metrics precision and recall and which is specifically

used when the dataset is imbalanced. The model has a large predictive power if the F1-score is near 1.0.

Precision is the ratio of correctly classified predicted abnormal observations of all the observations in the predicted class.

$$Precision = \frac{TP}{TP + FP} \quad (12)$$

Recall is the ratio of correctly predicted abnormal observations of all observations in the actual class.

$$Recall = \frac{TP}{TP + FN} \quad (13)$$

Hence, the F1-score is calculated using the following equation:

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (14)$$

Where: TP= True Positive; FP=False Positive; TN= True Negative; FN=False Negative.

## VI. RESULTS

To evaluate our model, we leverage the Python deep learning framework Pytorch [17]. We train and evaluate them on NVIDIA® Tesla® V100S with 32 GB HBM2 memory.

### A. Model Configuration & Hyperparameter tuning

As presented in Table II, for CAN-BERT, we have chosen the total number of transformer encoder layers as 4. In each transformer layer, the position-wise feed forward network is composed of two dense layers where the first one projects  $d=256$  dimensional CAN identifier embedding into  $d_{ff}=512$  dimensional space, followed by a ReLU activation. The second dense layer maps back the 512-dimensional vector into the  $d$ -dimensional space. For training, we use a batch size of 32, a learning rate of 0.001 and the Adam optimizer [18] with its default parameters  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . To avoid overfitting, we employ the same dropout of  $P_{drop}=0.1$  for all dropout layers in our network. Moreover, we apply early stopping for a total number of 200 epochs and a patience of 10 epochs as a form of regularization used to avoid overfitting.

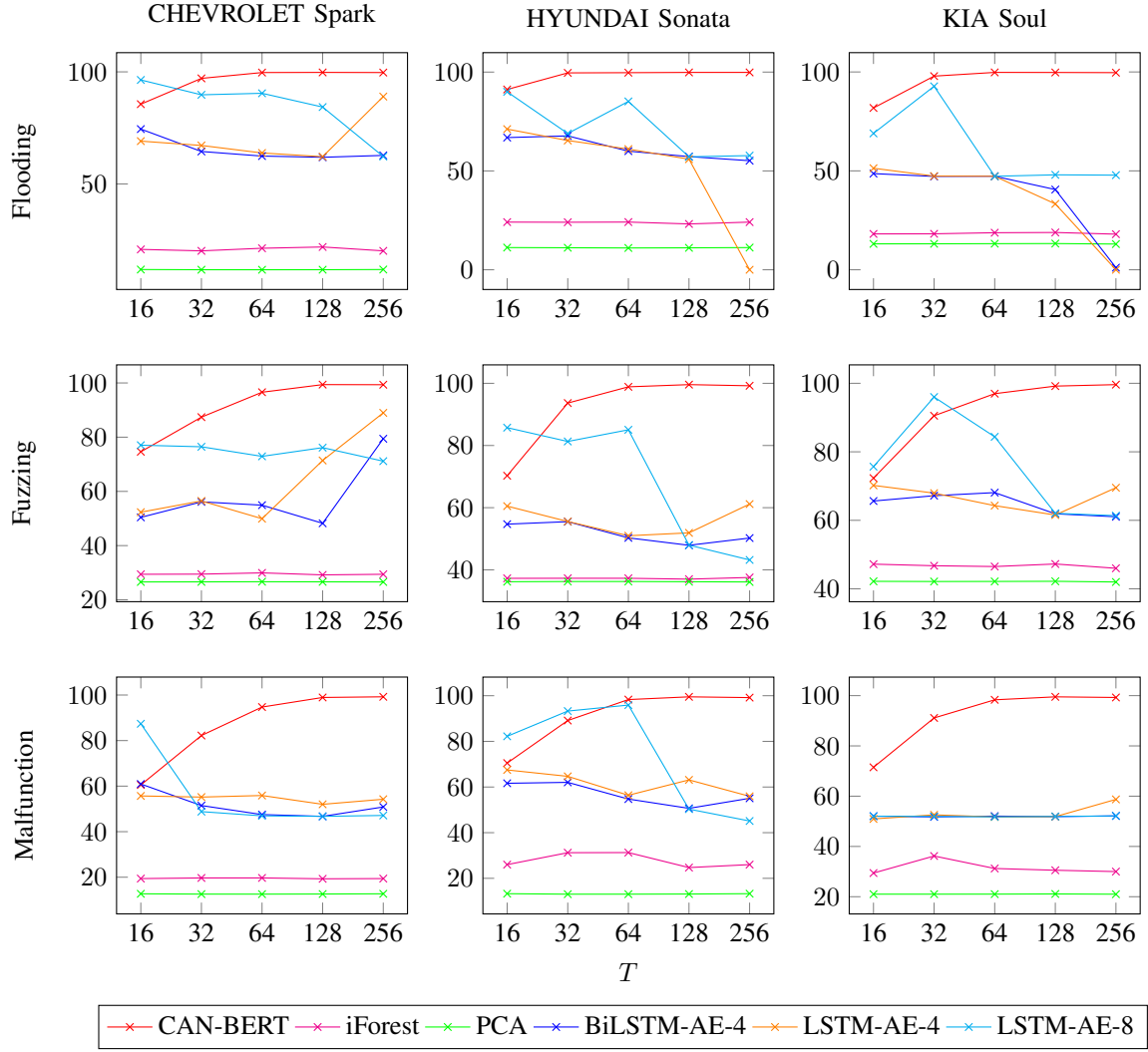


Fig. 6. Comparison of the CAN-BERT model with other anomaly detection baselines using the F1-score percentage metric, for different message injection attacks applied on different car models w.r.t sequence length  $T$ .

TABLE II  
CAN-BERT MODEL CONFIGURATION

Parameter	Value
$N$	4
$d_{model}$	256
$d_{ff}$	512
$h$	1
$P_{drop}$	0.1
$m$	0.45
# Candidates	5
Optimizer	Adam
Adam $\beta_1$	0.9
Adam $\beta_2$	0.999
Learning rate	0.001
Batch size	32
# Epochs	200
Patience	10

The hyperparameters, including the ratio of masked CAN IDs in a sequence  $m$ , and  $h$  the number of attention heads are

tuned based on a validation set for the three car types and the different message injection attacks. As seen in Figure 5, raising the ratios of masked CAN IDs in the sequences from 0.1 to 0.45 somewhat improves F1 scores, however increasing the ratios further degrades performance, as is the case for  $m=0.65$ . Furthermore, the model performance is relatively stable by setting different attention head  $h \in \{1, 2, 4, 8\}$  values for each mask ratio  $m \in \{0.15, 0.3, 0.45, 0.6\}$ . Therefore, in our in-vehicle intrusion detection use case, a single attention head is sufficient to detect different types of intrusions. Note that, in our experiments, we use the same ratio of masked CAN IDS  $m=0.45$  and  $h=1$  for both training and inference phases.

### B. Model Accuracy

As seen in Figure 6, we compare performance of the CAN-BERT model with other baselines approaches for different sequence length  $T$  using the F1-score metric. In fact, we varied the sequence length among values of 16, 32, 64, 128

and 256 CAN IDs per sequence in the experiments. If our approaches could identify a message injection attack in a shorter sequence length, it would be more advantageous in a practical situation. The traditional machine learning algorithms such as Isolation Forest (iForest), and Principal Component Analysis (PCA) perform poorly and maintain the same F1-score metric w.r.t sequence length. Because these models presume small datasets with a limited number of features, they fail to discover abnormalities in high-dimensional datasets. Because of this, a significant fraction of irrelevant features may effectively disguise the underlying abnormalities in the input data, resulting in poor anomaly identification performance when dealing with large input dimensions. Meanwhile, both deep learning based models autoencoder (AE) and CAN-BERT outperformed the traditional anomaly detection models over different window sizes. However, when the length of the CAN ID sequence is increased, both models performed oppositely. In contrast to the baseline models, our suggested model, CAN-BERT, significantly outperforms them by huge margins and obtains respectable F1 scores  $\in [0.85, 0.99]$ , demonstrating the usefulness of using BERT-based models to capture the patterns of CAN ID sequences when  $T \geq 32$ . However, on short sequence length as is the case for  $T = 16$ , the model performs modestly with F1-score  $\in [0.6, 0.9]$  for different kind of attacks. These experiments, therefore, reveal that by using self-supervised training tasks, CAN-BERT can effectively model medium to long normal CAN ID sequences and accurately detect anomalous sequences.

### C. Model Complexity

From a practical point of view, we must assess not only the classification performance but also the model complexity to check if the model's ability for real-time in-vehicle intrusion detection in CAN networks. Therefore, we assessed the inference time per sample as well as the number of parameters for the CAN-BERT model w.r.t different car types. As depicted in

TABLE III  
CAN-BERT MODEL COMPLEXITY

Vehicle	Features	Values
CHEVROLET Spark	Number of Parameters	2,937,422
	Model Size (MB)	[20, 70]
	Inference Time (ms)	[0.8, 3.1]
HYUNDAI Sonata	Number of Parameters	3,149,291
	Model Size (MB)	[20, 70]
	Inference Time (ms)	[0.8, 3.5]
KIA Soul	Number of Parameters	3,163,142
	Model Size (MB)	[20, 70]
	Inference Time (ms)	[0.8, 3.8]

Table III, the intrusion detection inference time varies between 0.8 and 3.1 ms w.r.t CAN ID sequence length. Hence, when considering a sequence length of 32 CAN IDs, our model detects an intrusion in 0.9 to 1 ms, which is suitable for real-time detection. Furthermore, having a size between 20MB and 70 MB and a number of parameters ranging between 2 to 3 millions, our model can be either deployed in performant ECU or even on a cloud server wirelessly connected to the vehicle.

## VII. CONCLUSION

Identification of intrusions within the vehicle is critical for defending it against malicious cyberattacks. In this paper, we suggest CAN-BERT, a self-supervised intrusion detection system based on BERT model, for in-vehicle intrusion detection. Experimental results on benchmark datasets for different CAN ID sequence length have shown that CAN-BERT surpasses state-of-the-art techniques for CAN ID sequence anomaly detection with an F1-score ranging between 0.81 and 0.99 for different type of attacks and is appropriate for real-time detection with an inference time ranging between 0.8 and 3 ms w.r.t CAN ID sequence length. For future work, we aim to deploy our model on embedded electronic control units (ECU) and test the model efficiency in a real vehicle environment.

## REFERENCES

- [1] Matheus, Kirsten, and Thomas Königseder. Automotive ethernet. Cambridge University Press, 2021.
- [2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
- [3] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
- [4] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).
- [5] Nam, Minki, Seungyoung Park, and Duk Soo Kim. "Intrusion detection method using bi-directional GPT for in-vehicle controller area networks." IEEE Access 9 (2021): 124931-124944.
- [6] Li, Bai, et al. "How is BERT surprised? Layerwise detection of linguistic anomalies." arXiv preprint arXiv:2105.07452 (2021).
- [7] Guo, Haixuan, Shuhan Yuan, and Xintao Wu. "Logbert: Log anomaly detection via bert." 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021.
- [8] Lee, Yookyung, Jina Kim, and Pilsung Kang. "LAnoBERT: System Log Anomaly Detection based on BERT Masked Language Model." arXiv preprint arXiv:2111.09564 (2021).
- [9] Le, Van-Hoang, and Hongyu Zhang. "Log-based anomaly detection without log parsing." 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2021.
- [10] Yu, Keping, et al. "Securing critical infrastructures: Deep-Learning-Based threat detection in IIoT." IEEE Communications Magazine 59.10 (2021): 76-82.
- [11] Dang, Weixia, et al. "TS-Bert: Time Series Anomaly Detection via Pre-training Model Bert." International Conference on Computational Science. Springer, Cham, 2021.
- [12] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. 2008.
- [13] Hyunjae Kang, Byung Il Kwak, Young Hun Lee, Haneol Lee, Hwejae Lee, Huy Kang Kim, February 3, 2021, "Car Hacking: Attack & Defense Challenge 2020 Dataset", IEEE Dataport, doi: <https://dx.doi.org/10.21227/qvr7-n418>
- [14] Abdi, Hervé, and Lynne J. Williams. "Principal component analysis." Wiley interdisciplinary reviews: computational statistics 2.4 (2010): 433-459.
- [15] Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation-based anomaly detection." ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012): 3.
- [16] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." nature 323.6088 (1986): 533-536.
- [17] Pytorch framework. <https://pytorch.org/>.
- [18] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
- [19] Hanselmann, Markus, et al. "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data." Ieee Access 8 (2020): 58194-58205.
- [20] Song, Hyun Min, Jiyoung Woo, and Huy Kang Kim. "In-vehicle network intrusion detection using deep convolutional neural network." Vehicular Communications 21 (2020): 100198.