Intrusion Detection Method Using Bi-Directional GPT for In-Vehicle Controller Area Networks

MINKI NAM1, SEUNGYOUNG PARK2, (Senior Member, IEEE), AND DUK SOO KIM3

1Interdisciplinary Graduate Program in BIT Medical Convergence, Kangwon National University, Chuncheon 24341, Republic of Korea
2Department of Electrical and Electronics Engineering, Kangwon National University, Chuncheon 24341, Republic of Korea
3Autocrypt Company Ltd., Seoul 07241, Republic of Korea

Corresponding author: Seungyoung Park (s.young.park@kangwon.ac.kr)

This work was supported by Autocrypt Company Ltd., Republic of Korea, the Ministry of Science and ICT (MSIT), Republic of Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Promotion (IITP) under Grant IITP-2021-2018-0-01433, and Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) through the Smart Farm Innovation Technology Development Program funded by the Ministry of Agriculture, Food and Rural Affairs (MAFRA) under Grant 421040-04.

ABSTRACT The controller area network (CAN) bus protocol is exposed to threats from various attacks because it is designed without consideration of security. In a normal vehicle operation situation, controllers connected to a CAN bus transmit periodic and nonperiodic signals. Thus, if a CAN identifier (ID) sequence is configured by collecting the identifiers of CAN signals in their order of occurrence, it will have a certain pattern. However, if only a very small number of attack IDs are included in a CAN ID sequence, it will be difficult to detect the corresponding pattern change. Thus, a detection method that is different from the conventional one is required to detect such attacks. Since a CAN ID sequence can be regarded as a sentence consisting of words in the form of CAN IDs, a generative pretrained transformer (GPT) model can learn the pattern of a normal CAN ID sequence. Therefore, such a model is expected to be able to detect CAN ID sequences that contain a very small number of attack IDs better than the existing long short-term memory (LSTM)-based method. In this paper, we propose an intrusion detection model that combines two GPT networks in a bi-directional manner to allow both past and future CAN IDs (relative to the time of detection) to be used. The proposed model is trained to minimize the negative log-likelihood (NLL) value of the bi-directional GPT network for a normal sequence. When the NLL value for a CAN ID sequence is larger than a prespecified threshold, it is deemed an intrusion. The proposed model outperforms a single unidirectional GPT model with the same degree of complexity as other existing LSTM-based models because the bi-directional structure of the proposed model maintains the estimation performance for most CAN IDs, regardless of their positions in the sequence.

INDEX TERMS Intrusion detection, generative pretrained transformer, GPT, controller area network, CAN, CAN ID, in-vehicle network, negative log-likelihood, NLL, spoofing attack.

I. INTRODUCTION Controller area network (CAN) bus communication is designed for in-vehicle communication. Specifically, an arbitrary electrical control unit (ECU) can broadcast a CAN data frame containing a CAN identifier (ID) and a message from any device connected to the bus [1]. As attacks against CAN bus communication have become more advanced and intelligent, increasingly sophisticated defense techniques against them have been proposed [2]–[6]. Existing methods detect the pattern changes caused by an attack after learning the normal pattern of a CAN ID sequence, which is composed of CAN IDs extracted only from CAN data frames. Hence, if the target sequence for detection contains only a very small number of attack IDs, the difference with respect to the normal pattern will be very small, making it difficult to detect. Therefore, a new detection method is required to detect such attacks.

For machine translation, long short-term memory (LSTM) models are commonly used because this task involves sequence data composed of words [7]. However, these models have the problem that the translation quality degrades in general as the sequence becomes longer. To solve this problem, the transformer network model was proposed [8]. In this
model, the words comprising a sentence and their positions in the sentence are converted into vectors through word token embedding and positional embedding, respectively, and they are processed through a multihead attention structure, thus solving the problem of performance degradation in the translation of long sentences.

Employing the decoder structure of the transformer network, the sentence generation technique known as generative pretrained transformer (GPT) was proposed by OpenAI [9], [10]. GPT predicts words that may appear at the current time point based on the words generated at the previous time points and then applies the same procedure again to predict the word that is to appear at the next time point. Thus, sentences can be generated using this autoregressive method. In other words, GPT has an excellent advantage in predicting the words that will appear after given words. GPT was trained for sentence generation using text data collected from an enormous corpus of 40 GB, which is called WebText, and the internet as the training data. It was shown that high-quality poems in English could be created using the GPT-2 model with 774 million parameters, and emotional reactions to/from readers were drawn [11]. In [12], it was shown that poems could be created in various languages, including English, Spanish, Ukrainian, Hindi, Bengali, and Assamese, also using a model of 774 million parameters, and they were difficult to distinguish from poems written by humans.

Since an ECU may send CAN data frames containing CAN IDs either regularly or irregularly, a CAN ID sequence will have a certain pattern that is difficult for humans to understand. If CAN IDs are treated as words, a CAN ID sequence can be regarded as a sentence composed of words in the form of CAN IDs. Hence, a GPT model can learn the normal patterns of CAN ID sequences. Considering the excellent sentence generation performance of GPT, it is expected to be able to detect even the fine pattern changes in a CAN ID sequence caused by a small number of attack IDs relatively well. Nevertheless, recalling that GPT will predict the CAN ID at the current time point using the previously generated CAN IDs, it may have difficulty generating predictions in the initial period due to the small number of CAN IDs to be used for prediction. To alleviate this problem, this paper proposes a CAN ID intrusion detection method that combines two GPT networks in a bi-directional manner [13].

Our research contributes the following findings:
1) Combining two GPT networks in a bi-directional manner is shown to significantly improve the detection performance compared to that of a single uni-directional GPT network with the same degree of complexity, even when the target sequence for detection contains a very small number of attack IDs.
2) The proposed method outperforms other existing methods, including a structure in which two LSTM networks are bi-directionally combined, for various attack types. Moreover, the performance shows further improvement with an increasing amount of training data compared to other existing methods.

This paper is organized as follows. Section II discusses the composition and security vulnerabilities of the CAN bus protocol and explains the overall CAN data frame structure, including the CAN IDs to be used for detection. Section III introduces existing methods of detecting abnormal patterns of CAN traffic in in-vehicle networks. Section IV describes the structure of the GPT model to be used in the proposed intrusion detection method. Section V proposes the bi-directionally combined GPT structure for detecting anomalies in a CAN ID sequence and explains the corresponding intrusion detection method. Section VI describes the experimental setup and conditions, evaluates the performance of the proposed method in this experimental environment and compares it with the performance of other methods. Finally, Section VII presents the conclusion.

II. OVERVIEW OF CAN BUS PROTOCOL

The CAN bus protocol is a standard communication protocol designed for efficient communication between ECUs without a host computer in the vehicle. Developed in 1983 by Bosch, the CAN bus protocol was established as the ISO 11898 standard in 1993 because of its simple yet efficient structure [1]. A CAN bus operates by means of a broadcasting method such that when one device transmits a message, every device connected to the bus can receive it. At this time, a CAN data frame composed of a CAN ID and the message is transmitted, and arbitrary devices connected to the CAN bus receive the CAN data frame corresponding to a specific CAN ID. Fig. 1 shows the CAN data frame structure used in the CAN bus protocol. The frame structure is composed of a start-of-frame (SOF) delimiter, an identifier (ID) field, a remote transmission request (RTR), a control field, a data
field, a cyclic redundancy code (CRC), an acknowledgment (ACK), and an end-of-frame (EOF) delimiter. For a CAN ID, 11 bits of the ID field in the data frame are used. Hence, it can support up to $2^{11}$ CAN IDs, from ‘0 × 000’ to ‘0 × 7FF’.

Because many devices sharing one physical bus may start a transmission on the bus while it is in an idle state, message arbitration (the process by which two or more devices agree on which is to use the bus) is of great importance for data transmission. The arbitration process is performed based on the CAN ID as follows. Consider the case in which two or more devices transmit CAN frames simultaneously. In this case, the transmitting devices monitor the bus during the period in which the CAN ID is transmitted while they are sending. If a device detects a dominant level when it is sending a recessive level itself, it refrains from transmission, switches to the receiving mode, and waits until the bus returns to the idle state. As a result, the transmission priority is determined by the CAN ID of the data frame, with a lower ID having a higher priority for transmission.

Unfortunately, the CAN bus protocol was developed without encryption or authentication features because no consideration was given to security at the time of its development [14]. Since many devices, such as Bluetooth devices, 3rd generation/4th generation (3G/4G) devices, Wi-Fi devices, wireless communication sensors, global positioning system (GPS) receivers, and vehicle controllers, as well as ECUs, may be connected in parallel to a single physical bus without security considerations, ECUs can be easily attacked from the outside [15]. As an example, Fig. 2 shows a situation in which ECUs for controlling speed and direction, which are critical to safety, are connected through one CAN bus along with external devices that control universal serial bus (USB) and Wi-Fi communication. If an attacker can access the CAN bus by gaining control of the devices responsible for external communication, he or she can interfere with the normal operation of the ECUs by sending malicious messages or saving normal frames and resending them. Moreover, some ECUs can be shut down by sending repetitive error messages, and intentional accidents can be caused by changing the information about the driving direction and velocity of a vehicle [16]. Such attacks are possible because the CAN data frame does not include the sender information and thus is easily spoofed.

### III. RELATED WORK

This section introduces various existing methods of detecting abnormal patterns of CAN traffic in in-vehicle networks. Reference [2] proposed a method that generates a first-order transition matrix for the CAN IDs constituting a CAN ID sequence and identifies the sequence as an attack if a CAN ID transition is not observed in the training period. However, this method is known to be vulnerable to replay attacks because attacks containing transitions related to a CAN ID that occurs frequently are not detected well.

A method of detecting whether a CAN ID sequence is abnormal using a generative adversarial network (GAN), which consists of a generator and a discriminator, after converting the CAN ID sequence into a binary image has been suggested [3]. The goal of the generator is to artificially create fake images that could be easily mistaken for real images, while the goal of the discriminator is to identify which images it receives have been artificially created [17]. This method demonstrated its ability to detect a CAN ID sequence that includes an attack by employing a discriminator trained using only normal CAN ID sequences. As an extension to this research, a convolutional neural network (CNN)-based intrusion method was proposed in which the CAN ID sequences were converted into binary images and the CNN was trained using both normal and attack CAN ID sequence images in a supervised manner [4].

Reference [5] suggested a method of detecting an attack by measuring the entropy of each bit comprising a CAN ID for a certain period. It showed its ability to detect attacks by sensing the change in the entropy for each CAN ID bit caused by an attacker.
by an injection attack. However, when the number of attacks during the measurement period is small, it is difficult to detect the attack behavior because the entropy change is negligible.

Reference [6] proposed a method of detecting attacks using a forward-direction prediction technique based on an LSTM network. Specifically, it predicts the log probability of the CAN ID appearing immediately after a given CAN ID sequence. After performing this process sequentially for a certain period, it identifies an attack by comparing the sum of the log probability values against a threshold.

A method of detecting whether an attack is performed through a given CAN ID using an LSTM-based autoencoder has been suggested [18]. The main idea of this method is to create a reconstructed time series of messages in CAN data frames for each CAN ID that minimizes the reconstruction error. This method demonstrated its ability to detect attacks by sensing the change in the Mahalanobis distance between the original and reconstructed time series of messages from the monitored CAN ID. However, considering the fact that messages from different CAN IDs affect each other interactively, the detection performance would be limited because this method considers messages from only one target CAN ID.

Reference [19] proposed a method of detecting attacks by employing a time-series analysis technique to capture the deterministic behavior of CAN traffic dynamics. To accomplish this, in this method, the CAN traffic is modeled as a time series of bytes extracted from the messages of consecutive CAN data frames. After the squared weighted Euclidean distance from the centroid, which is determined in advance using the normal traffic, is calculated for a given CAN traffic value, it identifies the attack by comparing the distance with the prespecified threshold. However, when the number of attacks is small in the target CAN traffic, it is difficult to detect the attack because its distance from the centroid would not be sufficiently large.

IV. DESCRIPTION OF GPT NETWORK

A typical machine translation tool uses a sequence-to-sequence model composed of an LSTM-based encoder and decoder. Specifically, the sentence to be translated is divided into word tokens, which are converted into word embedding vectors. Then, these tokens are sequentially provided to the encoder and compressed into a single-vector representation. Unfortunately, this structure causes some loss of information in the process of compressing a sentence into a single-vector representation. To mitigate this problem, an attention mechanism that refers back to the entire sentence to be translated as input into the encoder at each point in time at which the decoder predicts the output word has been proposed [20]. More specifically, in the attention mechanism, an attention function is employed to allocate higher weights to the words input to the encoder that are strongly related to the given output word to be predicted at the decoder. However, if the encoder and decoder are configured based on the LSTM structure, the translation performance will still be limited for long sentences even if this attention mechanism is used. To overcome this problem, the transformer method, which uses only an attention structure without LSTM, has been proposed [8].

GPT is a sentence generation method developed by OpenAI, a U.S. nonprofit artificial intelligence (AI) research...
institute, using the decoder structure from the transformer model [9]. Specifically, it is an autoregressive model using a masked self-attention structure in which the previous predicted output word is employed as the next input word during the sentence generation process because it has a good next-word prediction ability based on given input words [21].

Fig. 3 shows the details of the sentence generation process using the GPT network in an autoregressive manner. When the word ‘<s>’, representing the start (or end) of a sentence, is provided to the GPT network, the probabilities

$$P(·|<s>)$$

for all words that can occur after (or before) ‘<s>’ are estimated to guess the word that will appear next. To determine the next word, a top-k sampling technique is employed [10]. In top-k sampling, the probability mass function (PMF) is redistributed over the k most probable tokens, and the next word is sampled from those tokens in accordance with the redistributed PMF. For example, if the word ‘A’ is one of the top k tokens and is chosen in accordance with their redistributed PMF, it is provided to the GPT network. Then, the GPT network estimates the probabilities

$$P(·|<s>A)$$

for all words that can occur after (or before) ‘<s>A’ to guess the next word. For example, if the word ‘robot’ is one of the top k tokens and is chosen, it is, in turn, provided to the GPT network. This process is repeated until the word ‘<s>’, indicating the end (or start) of the sentence, is selected.

Fig. 4 shows the concrete structure of the GPT network for estimating the probability of a given sentence. It can be seen from this figure that the GPT network is composed of a stacked word and positional embedding layer, G transformer decoder blocks, a linear layer, and a softmax layer. In particular, each transformer decoder block is composed of a masked multihead self-attention layer, a layer normalization layer, a feedforward layer, and a second layer normalization layer. The specific operations performed by these components are explained as follows.

A. WORD AND POSITIONAL EMBEDDING LAYER

First, consider the case in which a sentence composed of L word tokens, expressed as

$$x = [x_0 \cdots x_{(L-1)}]^T,$$  \hspace{1cm} (1)

is provided to the GPT network and assume that the total number of word tokens that can be generated is K. Under this assumption, $$x_i$$ is an integer that satisfies

$$0 \leq x_i \leq (K - 1).$$

Because the GPT network predicts the next word corresponding to the input words [9], it will estimate the probability of a sentence of the form

$$\tilde{x} = [x_1 \cdots x_L]^T$$  \hspace{1cm} (2)

if x is provided. Note that $$\tilde{x}$$ is the same sentence with a delay of one word token relative to x.

In this configuration, each word token $$x_i$$ is converted into an E-dimensional word embedding vector. Then, it becomes an E-dimensional vector $$y_i$$ after the addition of an E-dimensional positional vector that depends on the position of the word in the sentence [8]. Specifically, for the positional encoding vector, the vector corresponding to the position of the given word token is selected from a vector set composed of L different vectors [9]. Consequently, the sentence x is converted into a matrix with dimensions of E x L,

$$Y = [y_0 \cdots y_{(L-1)}].$$  \hspace{1cm} (3)

Then, these L vectors are simultaneously provided to the masked multihead self-attention layer, which is composed of H heads.

B. MASKED MULTIHEAD SELF-ATTENTION LAYER

For the $$h^{th}$$ head of the masked multihead self-attention layer, Y is transformed into a query matrix

$$Q^{(h)} = (W_Q^{(h)})^T Y,$$

a key matrix

$$K^{(h)} = (W_K^{(h)})^T Y,$$
and a value matrix

$$V^{(h)} = \left( W^{(h)}_V \right)^T Y,$$

where $W^{(h)}_V$ are all $E \times p$ matrices, with $p$ being an integer that satisfies $p \times H = E$. Then, these matrices are used to construct an attention value matrix with dimensions of $p \times L$ as follows:

$$Z^{(h)} = \begin{bmatrix} z^{(h)}_0 & \cdots & z^{(h)}_{(L-1)} \end{bmatrix} = V^{(h)} \cdot \text{softmax} \left( \frac{1}{\sqrt{p}} \left( Q^{(h)} \right)^T K^{(h)} + M \right),$$

(4)

where $M$ is a mask matrix with dimensions of $L \times L$. In (4), the $(i,j)^{th}$ element of $M$, $m_{i,j}$, is defined as

$$m_{i,j} = \begin{cases} 0 & \text{if } j \leq i, \\ \infty & \text{if } j > i. \end{cases}$$

(5)

In addition, for the matrix input

$$A = \left( \frac{1}{\sqrt{p}} \left( Q^{(h)} \right)^T K^{(h)} + M \right)$$

with dimensions of $L \times L$, softmax($\cdot$) is given as

$$\text{softmax}(A) = \frac{\exp(a_{i,0})}{\sum_{l=0}^{L-1} \exp(a_{i,l})}, \ldots, \frac{\exp(a_{i,(L-1)})}{\sum_{l=0}^{L-1} \exp(a_{i,(L-1)})},$$

where $a_{i,j}$ is the $(i,j)^{th}$ element of $A$. Due to the mask matrix $M$, the attention value vector at point $l$, $z_l^{(h)}$ (that is, the $l^{th}$ column vector in the attention value matrix in (4)), can use only the attention for the input word tokens

$$\{x_l^{(l)}\}^{l=0}_{l=0}$$

up to the corresponding point and cannot use the attention for the future input word tokens

$$\{x_l^{(L-1)}\}^{(L-1)}_{l=(l+1)}.$$

The $H$ attention value matrices $\{Z^{(h)}\}^{(H-1)}_{h=0}$ that were generated from the masked multihead self-attention layer are concatenated in a row to produce a matrix with dimensions of $E \times L$, which is then provided to the linear layer to produce a matrix with dimensions of $E \times L$ as follows:

$$T = W^T_O \begin{bmatrix} Z^{(0)} \\ \vdots \\ Z^{(H-1)} \end{bmatrix},$$

(6)

where $W_O$ is a matrix with dimensions of $E \times E$ that constitutes the linear layer.

C. FEEDFORWARD LAYER

The matrix $T'$ is produced by adding the residual input $Y$ [22] and applying layer normalization [23] as follows:

$$T' = \text{layernorm}(T + Y),$$

(7)

as shown in Fig. 4. This matrix is provided to the feedforward layer, which consists of a linear layer, an activation function, and a second linear layer, to produce a matrix with dimensions of $E \times L$ as follows:

$$T'' = W^T_{F1} \text{GELU} \left( W^T_{F0} T' \right),$$

(8)

where $W_{F0}$ and $W_{F1}$ are matrices with dimensions of $E \times 4E$ and $4E \times E$, respectively, and GELU($\cdot$) is the Gaussian error linear unit activation function [24].

The matrix $T''$ is then added to the residual input matrix $T'$ and subjected to layer normalization to produce a matrix with dimensions of $E \times L$, expressed as

$$P^{(0)} = \text{layernorm}(T'' + T'),$$

(9)

as the output of the first transformer decoder block.

D. SENTENCE PROBABILITY ESTIMATION

Since the GPT network has a stacked structure with $G$ transformer decoder blocks, the above process is repeated $G$ times, and a matrix with dimensions of $E \times L$, $P^{(G-1)}$ is finally produced, as shown in Fig. 4. The top linear layer takes the matrix $P^{(G-1)}$ as input and produces a matrix with dimensions of $K \times L$ as follows:

$$U = W^T_D P^{(G-1)} = [u_0 \cdots u_{(L-1)}],$$

(10)

where $W_D$ is a matrix with dimensions of $E \times K$ that constitutes the top linear layer. Using the column vector

$$u_l = [u_{0,l} \cdots u_{(K-1),l}]^T,$$

the softmax layer produces

$$\text{softmax}(u_l) = \left[ \frac{\exp(u_{0,l})}{\sum_{l=0}^{K-1} \exp(u_{0,l})}, \ldots, \frac{\exp(u_{K-1},l)}{\sum_{l=0}^{K-1} \exp(u_{K-1},l)} \right]^T.$$

(11)

Because i) the GPT network predicts the next word corresponding to the given input words as discussed in Section IV-A, and ii) $u_l$ is generated from the input word tokens

$$\{x_l^{(l)}\}^{l=0}_{l=0}$$

due to the introduction of the mask matrix $M$ in (4), (11) can be regarded as a conditional probability estimate for the word token $x_{(l+1)}$ given the input word tokens up to point $l$, expressed as

$$\hat{P}(x_{(l+1)} | \{x_l^{(l)}\}^{l=0}_{l=0}).$$
As a result, the estimated probability for \( \hat{x} \) in (2) becomes

\[
\hat{P}(\hat{x}) = \prod_{i=0}^{(L-1)} \hat{P}(x_{i+1} = y_{i+1} | x_i^{t=0}) ,
\]

where \( y_{i+1} \) is the ground truth for \( x_{i+1} \) [9].

V. PROPOSED METHOD FOR INTRUSION DETECTION

In this section, the proposed CAN bus intrusion detection method based on a bi-directional GPT network is explained. Each ID in the input CAN ID sequence is converted into an integer and provided to the bi-directional GPT network to evaluate the negative log-likelihood (NLL) value for the sequence, and whether it is an attack is determined by comparing it against a prespecified threshold value.

A. DESCRIPTION OF THE BI-DIRECTIONAL GPT NETWORK

Suppose that the CAN ID sequence has a length of \( L \) and comprises \( (K - 1) \) valid CAN IDs. In addition, suppose that for each CAN ID in the given CAN ID sequence, the CAN ID values are mapped to integers in ascending order from 0 to \( (K - 2) \), and CAN IDs that do not exist among the normal CAN signals are converted into the integer \( (K - 1) \). Consequently, the CAN ID sequence is expressed as

\[
x = [x_0 \cdots x_{(L-1)}],
\]

satisfying \( 0 \leq x_i \leq (K - 1) \).

Fig. 5 shows the bi-directional GPT network structure, which combines forward and backward GPT networks. Note that the forward and backward GPT networks are each composed of a stacked word and positional embedding layer and \( G \) transformer decoder blocks. The intrusion detection process for the given CAN ID sequence \( x \) using the bi-directional GPT network structure is explained as follows. The given CAN ID sequence \( x \) is converted into

\[
f = [x_0 \cdots x_{(L-2)}]^\top \quad (13)
\]

and

\[
b = [x_{(L-1)} \cdots x_1]^\top , \quad (14)
\]

which are then provided to the forward and backward GPT networks, respectively. The forward GPT network is provided with \( f \) and produces the following \( E \times (L - 1) \) matrix:

\[
F = [\hat{f}_0 \cdots \hat{f}_{(L-2)}] , \quad (15)
\]

where \( \hat{f}_l \) is the \( E \)-dimensional vector containing information related to

\[
\mathbb{P}(x_{l+1} | x_0, \cdots, x_l) ,
\]

as discussed in Section IV-D. Similarly, the backward GPT network uses \( b \) to produce the following \( E \times (L - 1) \) matrix:

\[
B = [\hat{b}_0 \cdots \hat{b}_{(L-2)}] , \quad (16)
\]

where \( \hat{b}_l \) is the \( E \)-dimensional vector containing information related to

\[
\mathbb{P}(x_{l-1} | x_{(L-1)}, \cdots, x_{(L-l)}) .
\]

As mentioned above, at early prediction points, the number of observed CAN IDs that can be used is small. To alleviate this problem, \( F \) and \( B \) are combined to produce a combined matrix with dimensions of \( 2E \times L \) as follows:

\[
C = [\mathbf{0}_E \ F \ 
\mathbf{G} \ 0_E]
\]

\[
= [c_0 \cdots c_{(L-1)}] , \quad (17)
\]

where \( \mathbf{G} \) is an exchange matrix to match the sequence of column vectors comprising \( B \) with the column vector sequence of \( F \) [25] and \( \mathbf{0}_E \) is an \( E \)-dimensional zero vector. The top linear layer takes \( C \) as input and produces a matrix with dimensions of \( K \times L \) as follows:

\[
U = W^\top C
\]

\[
= [u_0 \cdots u_{(L-1)}] , \quad (18)
\]

where \( W \) is a matrix with dimensions of \( 2E \times K \) that constitutes the top linear layer. Then, \( u_L \) is passed to the softmax layer. Considering that the output matrices of the forward and backward GPT networks are combined, as shown in (17), the softmax layer output can be regarded as the conditional probability estimate of \( x_l \) for all given CAN IDs in \( x \) except for itself, expressed as

\[
\hat{P}(x_l | x_{t=0}, \cdots, x_{t=0}, l \neq t) . \quad (19)
\]

Finally, the estimated probability for the CAN ID sequence \( x \) in (12) becomes

\[
\hat{P}(x) = \prod_{l=0}^{(L-1)} \hat{P}(x_l) = y_l \prod_{t=0}^{(L-1)} \hat{P}(x_l | x_{t=0}, \cdots, x_{t=0}, l \neq t) , \quad (20)
\]

where \( y_l \) is the ground-truth value for \( x_l \).
FIGURE 6. ECDF for the injection interval of attack CAN IDs.

B. DETECTION METHOD
Suppose that \( N \) normal CAN ID sequences have been collected for training. Using the collected training data, training is performed such that the output of the bi-directional GPT network should minimize the NLL loss function, defined as

\[
NLL = -\frac{1}{NL} \sum_{n=0}^{N-1} \sum_{l=0}^{L-1} \log \hat{P}(x^{(n)}_l = y^{(n)}_l \mid \{x^{(n)}_{l'}\}_{l'=0, l' \neq l}^{(L-1)}),
\]

where \( x^{(n)}_l \) is the \( l \)th CAN ID in the \( n \)th normal CAN ID sequence used for training and \( y^{(n)}_l \) is the ground-truth value for \( x^{(n)}_l \).

After training is completed, to detect whether a given CAN ID sequence \( x \) shows evidence of attack behavior, \( x \) is provided as input to the bi-directional GPT network to evaluate

\[
NLL_x = -\frac{1}{L} \sum_{l=0}^{L-1} \log \hat{P}(x_l = y_l \mid \{x_{l'}\}_{l'=0, l' \neq l}^{(L-1)}). \tag{21}
\]

Finally, an attack is identified if the value of \( NLL_x \) is greater than a prespecified threshold \( \Gamma \), that is, if

\[
NLL_x > \Gamma \tag{22}
\]

is satisfied.

VI. EXPERIMENTAL SETUP AND RESULTS
This section describes the composition of the attacks used in the experiments and compares the performance of the proposed method with the performance of existing methods under these experimental conditions.

A. EXPERIMENTAL SETTINGS AND PERFORMANCE METRICS
In this paper, we collected and used CAN bus signals from the 2020 Hyundai Avante CN7. A normal CAN bus signal comprises a total of 90 valid CAN IDs. For training purposes, the vehicle was driven around downtown for approximately 1.8 hours, and approximately 15,900,000 CAN ID sequences were collected unless otherwise stated. For evaluation purposes, attacks were conducted for approximately 0.34 hours, and approximately 3,300,000 CAN ID sequences were obtained. In the process of collecting evaluation data, flooding, spoofing, replay, and fuzzing attacks were conducted as attacks on the target vehicle. The detailed methods of conducting the attacks were as follows. In the flooding attacks, approximately 154,200 instances of CAN ID ‘0 × 000’, the ID with the highest priority, were injected into the CAN bus. In the spoofing attacks, 2 valid CAN IDs were selected from the group of suitable CAN IDs, and approximately 7,800 of them were injected. In the replay attacks, approximately 47,600 normal CAN bus signals were recorded for a set period of time, and they were then reinjected. In the fuzzing attacks, CAN IDs were randomly generated, and approximately 89,900 of them were injected. Table 1 summarizes the composition of the training and evaluation data.

Unless otherwise noted, the following settings were applied in the proposed method: CAN ID sequence length, \( L = 256 \); dimensionality of word and positional embedding vectors, \( E = 128 \); number of attention heads in the forward and backward GPT networks, \( H = 8 \); number of transformer decoder blocks in the forward and backward GPT networks, \( G = 6 \); and dropout probability in each layer, 0.1. Training
was performed using only training data consisting of the aforementioned normal CAN bus signals. In this process, the minibatch size was set to 32, and training was performed for 10 epochs using the adaptive moment estimation (Adam) optimization algorithm [26]. If one or more attack CAN IDs were identified to exist within a CAN ID sequence with a length of \( L \) in the evaluation process, this sequence was considered an attack sequence.

The true positive rate (TPR) and false positive rate (FPR) were used as metrics to evaluate the performance. The TPR is the ratio of the number of CAN ID sequences correctly determined to be attacks (i.e., true positives) to the total number of attack CAN ID sequences, and the FPR is the ratio of the number of CAN ID sequences falsely determined to be attacks (i.e., false positives) to the total number of normal CAN ID sequences. Furthermore, based on these performance metrics, we used the receiver operating characteristic (ROC) curve to visually illustrate the performance of intrusion detection systems utilizing various threshold values. The ROC curve is constructed by plotting the FPR and TPR values corresponding to each threshold value on the horizontal and vertical axes, respectively, of a two-dimensional graph. To compare the ROC performance of different methods, the area under the curve (AUC) was used. The AUC value was calculated by normalizing the area underneath the given ROC curve, resulting in an AUC value of 1 for perfect performance. A typical performance detector may exhibit various AUC values ranging between 0 and 1. Therefore, we can deduce that the higher the AUC value of the detector under consideration is, the higher its performance. In addition, we used the F-measure, the harmonic mean of precision and recall (i.e., TPR), which is defined as

\[
F\text{-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.
\]

where the precision is the ratio of the number of actual attack CAN ID sequences to the total number of CAN ID sequences identified as attacks. A higher F-measure value is considered to correspond to a higher detection capability.

B. STATISTICAL CHARACTERISTICS OF ATTACKS

Fig. 6 illustrates the empirical cumulative density function (ECDF) of the injection intervals of attack CAN IDs. For example, in the case of flooding attacks, the injection interval was set to 3, which means that one out of every three CAN data frames transmitted through the CAN bus originated from a flooding attack. For flooding, fuzzing, and replay attacks, we can see that most injection intervals of attack CAN IDs are below 10 due to the nature of these attacks. Therefore, these attacks are expected to be readily noticeable, leading to reasonable detection performance, as there will be a sufficient number of attack IDs even if the length of the CAN ID sequence, \( L \), is not large. In contrast, spoofing attacks are conducted by using only 2 valid CAN IDs, meaning that the injection interval of attack IDs is relatively large. Specifically, approximately 12% of the spoofing attacks had an injection...
interval of more than 100. Therefore, when \( L = 100 \), approximately 12% of spoofing attacks will produce only one attack ID within the corresponding CAN ID sequence.

Fig. 7 shows the ECDF regarding the number of attack IDs within one CAN ID sequence for \( L = 256 \). We can see that for spoofing attacks, very few attack IDs indeed exist within one CAN ID sequence, with a minimum of 1 to a maximum of 11. Specifically, in approximately 5% of all spoofing CAN ID sequences, 2 or fewer spoofing attacks are present. In other words, too few spoofing attacks appear in a substantial number of spoofing CAN ID sequences, which is expected to cause the detection performance to deteriorate for this type of attack.

**C. PERFORMANCE OF THE PROPOSED METHOD**

To determine the most suitable length \( L \) of a CAN ID sequence, Fig. 8 shows the ROC performance of the proposed method for spoofing attacks as a function of \( L \). As predicted in the previous subsection, when \( L = 64 \) or 128, the number of spoofing attacks within a single sequence is too small, meaning that the detection performance is low. On the other hand, when \( L = 256 \), the detection performance is enhanced because two or more spoofing attacks are present per sequence in most cases. Furthermore, the AUC performance when \( L = 256 \) is higher by approximately 6.9% and 1.5%, respectively, than the cases of \( L = 64 \) and \( L = 128 \). Therefore, unless otherwise noted, \( L = 256 \) for all subsequent results.

Fig. 9 shows the ECDF of the NLL values for attacks with \( L = 256 \). In the cases of flooding, fuzzing, and replay attacks, the NLL value tends to be very large compared to its normal level because there are so many attack IDs within one CAN ID sequence. However, since spoofing attacks produce a relatively small number of attack IDs, the corresponding NLL value is lower.

**D. PERFORMANCE COMPARISONS**

In this subsection, we compare the performance of the proposed method with that of existing intrusion detection methods. The first intrusion detection method considered for comparison is the bi-directional Markov method, in which second-order Markov-chain models are combined in a bi-directional manner [27]. In this method, the training data are used to estimate the second-order transition probability of a CAN ID sequence in the forward and backward directions, and this value can be used to calculate the log probability of the evaluated sequence to determine the attack status.

To investigate the performance gain due to the bi-directional structure of the proposed method, in the second considered method, a GPT network is applied only in the forward direction (that is, a uni-directional GPT network) [9]. Because this method uses only a one-way GPT network, the number of transformer decoder blocks in the uni-directional GPT network, \( G \), is set to 12 to maintain a complexity similar to that of the proposed method.

To compare the performance achieved with a GPT model to that of an LSTM model, the third considered method is a bi-directional LSTM method using two LSTM networks instead of GPT networks as used in the proposed method. In this method, word embedding vectors of the same dimensionality \( (E = 128) \) as in the proposed method are passed to the forward and backward LSTM networks, and each LSTM network with layer normalization is composed of a stack.
6 LSTM cells with hidden state and cell state dimensions of 128.

For comparison with the performance of other existing methods, the GAN-based method proposed in [3] and the LSTM-based prediction method proposed in [6] were selected as the fourth and fifth methods, respectively. To ensure fair comparisons, the GAN was trained using binary images converted from CAN ID sequences with a length of 256. In this method, a CAN ID sequence was identified as an attack sequence if the discriminator output was less than a prespecified threshold. The LSTM-based prediction method, as introduced in Section III, predicts the log probability of the next CAN ID for a given CAN ID sequence by means of an LSTM network. To ensure fair comparisons, a word embedding vector of the same dimensionality ($E = 128$) as in the proposed method was employed, and the LSTM network with layer normalization was composed of a stack of 12 LSTM cells with hidden state and cell state dimensions of 128. The LSTM network was used to predict the next CAN ID for an input CAN ID sequence with a length of 256. Accordingly, in this method, an attack was identified by comparing the sum of the log probabilities of 256 consecutive CAN IDs against a prespecified threshold [6].

Fig. 10 compares the ROC performance for spoofing attacks of the intrusion detection methods with $L = 256$. As expected, the AUC performance of the proposed method is improved compared to the other methods. In particular, compared to the uni-directional GPT model, the proposed model combining GPT networks in both the forward and backward directions can achieve higher performance with the same degree of complexity. Additionally, the GAN-based method fails to detect spoofing attacks because the binary images converted from CAN ID sequences containing only a small number of spoofing attacks look very similar to those corresponding to normal CAN ID sequences.

Table 2 compares the TPR performance of the different intrusion detection methods for spoofing attacks. The proposed method also shows an increase in the TPR compared to the other methods. For example, at an FPR of 0.5%, the TPR performance of the proposed method is improved by approximately 207.4% compared to that of the bi-directional Markov method.

Table 3 summarizes the performance of the different intrusion detection methods in terms of the false negative rate (FNR) and the F-measure at an FPR of 0.5%. Here, the FNR is the ratio of the number of attacks falsely determined to be normal to the total number of actual attacks (i.e., $1 - \text{TPR}$), and a lower FNR value corresponds to better performance. We can see that both the FNR and F-measure of the proposed method are universally improved compared to those of the other methods regardless of the attack type.
In particular, the performance improvement for spoofing attacks is significant. This means that the proposed method can detect slight changes in the pattern of a CAN ID sequence containing only a small number of attacks. The reason is explained as follows.

Fig. 11 shows the mean NLL values for the CAN IDs in a normal CAN ID sequence with $L = 256$. Note that the mean NLL values should be lower for normal CAN ID sequences. This figure indicates that the mean NLL performance of the uni-directional GPT method tends to be degraded at earlier positions in a sequence because the NLLs of CAN IDs at earlier positions are estimated using fewer CAN IDs. On the other hand, in the proposed method, the NLL performance is maintained for all CAN IDs regardless of their positions. The
TABLE 3. Performance comparison for various attacks.

|               | Bi-directional Markov | Uni-directional GPT [9] | Bi-directional LSTM | LSTM Pred. [6] | GAN [3] | Proposed |
|---------------|------------------------|-------------------------|---------------------|----------------|---------|----------|
|               | FNR F-measure          | FNR F-measure           | FNR F-measure       | FNR F-measure  | FNR F-measure | FNR F-measure |
| Flooding      | 4.41e-4 9.895e-1       | 2.39e-4 9.896e-1       | 1.17e-4 9.896e-1   | 2.88e-4 9.896e-1 | 5.31e-4 9.896e-1 | 7.882e-5 9.897e-1 |
| Spoofing      | 6.95e-1 4.855e-1       | 3.75e-1 7.547e-1       | 1.57e-1 8.995e-1   | 2.58e-1 8.366e-1 | 9.894e-1 1.29e-2 | 6.27e-2 9.524e-1 |
| Replay        | 3.24e-3 9.848e-1       | 1.45e-3 9.857e-1       | 9.06e-4 9.859e-1   | 1.04e-3 9.859e-1 | 2.54e-3 8.41e-1  | 7.057e-4 9.860e-1 |
| Fuzzing       | 1.63e-3 9.875e-1       | 4.92e-4 9.880e-1       | 1.52e-4 9.881e-1   | 8.221e-4 9.878e-1 | 2.603e-3 9.889e-1 | 1.189e-4 9.881e-1 |

FIGURE 12. F-measure performance against spoofing attacks as a function of the recording time length of the training data.

reason is that the NLL of a given CAN ID is estimated using both past and future CAN IDs. However, the performance for early and late positions is still slightly degraded because the first and last positions in the combined matrix in (17) are padded with zero values. Additionally, the bi-directional LSTM method shows a performance trend similar to that of the proposed method because it also employs a network structure combining two networks in a bi-directional manner. However, its overall performance is slightly lower. This implies that GPT itself is superior to LSTM for the prediction task. Consequently, the ability to achieve stable CAN ID estimation improves the overall detection performance.

To obtain all of the results discussed above, we evaluated the performance after training on 1.8 hours of recorded normal CAN ID sequences. To investigate the effect of the training data size on the performance, Fig. 12 shows the F-measure performance for spoofing attacks achieved with training data corresponding to different recording time lengths for an FPR of 0.5% and $L = 256$. From this figure, we see that the performance of the proposed method improves as the recording time length increases. Additionally, the proposed method leads to universally better performance regardless of the recording time. This demonstrates that the proposed method is still more efficient than the other methods when a large volume of recorded data is available for training.

VII. CONCLUSION

This paper has proposed a bi-directional GPT-based method for intrusion detection based on CAN ID sequences. This method can serve to identify an attack against the CAN bus protocol, which lacks security. Because the existing intrusion detection methods underperform in detecting a CAN ID sequence that contains very few attacks, this paper has suggested a model comprising two GPT networks connected bi-directionally. The proposed method computes the NLL value for a CAN ID sequence and compares it against a prespecified threshold to identify any attack. In experiments, the proposed bi-directional GPT network proved to achieve superior performance compared with a uni-directional GPT network of the same complexity. The reason for this performance improvement is that the bi-directional structure of the proposed method allows the prediction performance for each CAN ID to be maintained regardless of its position in the CAN ID sequence.

In this paper, we considered only the case in which malicious CAN packets are injected into the CAN bus. Thus, the proposed method was designed to detect the change in the
CAN ID sequence due to this injection. However, when an intended ECU is reprogrammed by an attacker, the message in a CAN data frame can be maliciously manipulated and sent without affecting the normal pattern of the ID CAN sequence. Therefore, in future work, it will be necessary to extend the proposed method to detect abnormal patterns in the messages.

REFERENCES

[1] H. Chen and J. Tian, “Research on the controller area network,” in Proc. Int. Conf. Netw. Digit. Soc., May 2009, pp. 251–254.

[2] M. Marchetti and D. Stabili, “Anomaly detection of CAN bus messages through analysis of ID sequences,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2017, pp. 1577–1583.

[3] E. Seo, H. M. Song, and H. K. Kim, “GIDS: GAN based intrusion detection system for in-vehicle network,” in Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST), Aug. 2018, pp. 1–6.

[4] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” Veh. Commun., no. 1, pp. 1–13, Jan. 2020.

[5] Q. Wang, Z. Lu, and G. Qu, “An entropy analysis based intrusion detection system for controller area network in vehicles,” in Proc. 31st IEEE Int. Syst. Chip Conf. (SOCC), Sep. 2018, pp. 90–95.

[6] A. K. Desta, S. Ohira, I. Arai, and K. Fujikawa, “ID sequence analysis for intrusion detection in the CAN bus using long short term memory networks,” in Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops), Mar. 2020, pp. 1–6.

[7] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” 2017, arXiv:1702.01923. [Online]. Available: http://arxiv.org/abs/1702.01923

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. Int. Conf. Neural Inf. Process. Syst., Long Beach, CA, USA, Dec. 2017, pp. 6000–6010.

[9] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, (2018). Improving Language Understanding by Generative Pre-Training. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, Language models are unsupervised multitask learners, 2019. [Online]. Available: http://www.persagen.com/files/misc/radford2019language.pdf

[11] B. Bena and J. Kalita, “Introducing aspects of creativity in auto-matic poetry generation,” 2020, arXiv:2002.02511. [Online]. Available: http://arxiv.org/abs/2002.02511

[12] D. W. Otter, J. R. Medina, and J. K. Kalita, “A survey of the usages of deep learning for natural language processing,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 2, pp. 604–624, Feb. 2021.

[13] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[14] C.-W. Lin and A. Sangiovanni-Vincentelli, “Cyber-security for the controller area network (CAN) communication protocol,” in Proc. Int. Conf. Cyber Secur., Dec. 2012, pp. 1–7.

[15] S. Woo, H. Jin Jo, and D. Hoon Lee, “A practical wireless attack on the connected car and security protocol for in-vehicle CAN,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[16] B. Groza and P.-S. Murvay, “Security solutions for the controller area network: Bringing authentication to in-vehicle networks,” IEEE Veh. Technol. Mag., vol. 13, no. 1, pp. 40–47, Mar. 2018.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.

[18] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero, “CAN2node: An anomaly detection system based on LSTM autoencoders for controller area network,” IEEE Trans. Netw. Service Manage., vol. 18, no. 2, pp. 1913–1924, Jun. 2021.

[19] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olofsson, “CASAD: CAN-aware stealthy-attack detection for in-vehicle networks,” 2019, arXiv:1909.08407. [Online]. Available: http://arxiv.org/abs/1909.08407

[20] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2014, arXiv:1409.0473. [Online]. Available: http://arxiv.org/abs/1409.0473

[21] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” 2019, arXiv:1906.08237. [Online]. Available: http://arxiv.org/abs/1906.08237

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” 2016, arXiv:1603.05027. [Online]. Available: http://arxiv.org/abs/1603.05027

[23] J. Lei Ba, J. R. Kiros, and G. Hinton, “Layer normalization,” 2016, arXiv:1607.06450. [Online]. Available: http://arxiv.org/abs/1607.06450

[24] D. Hendrycks and K. Gimpel, “Gaussian error linear units (GELUs),” 2016, arXiv:1606.08415. [Online]. Available: http://arxiv.org/abs/1606.08415

[25] A. K. Horn and C. R. Johnson, Matrix Analysis, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[26] R. A. Stover and C. R. Johnson, Matrix Analysis, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] N. Ye, Y. Zhang, and C. M. Borror, “Robustness of the Markov-chain model for cyber-attack detection,” IEEE Trans. Rel., vol. 5, no. 1, pp. 116–123, Mar. 2004.

MINKI NAM received the B.S. degree in electrical and electronics engineering from Kangwon National University, Chuncheon, Republic of Korea, in 2021, where he is currently pursuing the M.S. degree with the Interdisciplinary Graduate Program in BIT Medical Convergence. His research interests include deep learning and natural language processing.

SEUNGYOUNG PARK (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea University, Seoul, Republic of Korea, in 1997, 1999, and 2002, respectively. From April 2003 to December 2005, he was with Samsung Advanced Institute of Technology, Kiheung, Republic of Korea, where he was a Senior Engineer, working on several projects in the field of next-generation wireless mobile communications. From January 2006 to February 2007, he was with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, where he was a Postdoctoral Research Associate. Since March 2007, he has been with the Department of Electrical and Electronics Engineering, Kangwon National University, Chuncheon, Republic of Korea, where he is currently a Professor. His research interests include machine learning applications in controller area networks, internet security, and communications.

DUK SOO KIM received the B.S. and M.S. degrees in electrical engineering from POSTECH, Pohang, Republic of Korea, in 1997 and 1999, respectively. From 1999 to 2019, he led the development of security software and served as CSO for a security software company, Penta Security Systems Inc., Seoul, Republic of Korea, for a period of 20 years. From 2014 to 2018, he was the Chief Director of the APGC-Laboratory, POSTECH’s start-up incubation laboratory. He served as an Advisor to a machine learning company, XBrain Inc., Seoul, and was involved in the founding of a security-as-a-service company, Cloudbric Corporation, Seoul, in 2017, where he served as a Board Member, in 2018. Since 2019, he has been with an autonomous vehicle security technology firm, Autocrypt Company Ltd., Seoul, where he is currently a Co-Founder and COO. His research interests include detecting anomalies in automobile, mobility, and transport systems and protecting them from external threats.