electronics

MDPI

Article

Using Deep Learning Networks to Identify Cyber Attacks on Intrusion Detection for In-Vehicle Networks †

Hsiao-Chung Lin [1], Ping Wang [1,*], Kuo-Ming Chao [2], Wen-Hui Lin [1] and Jia-Hong Chen [1]

[1] Green Energy Technology Research Center, Faculty of Department of Information Management, Kun Shan University, Tainan 710303, Taiwan; fordlin@mail.ksu.edu.tw (H.-C.L.); linwh@mail.ksu.edu.tw (W.-H.L.); s106000750@g.ksu.edu.tw (J.-H.C.)

[2] Department of Computing & Informatics, Bournemouth University, Bournemouth BH12 5BB, UK; kchao@bournemouth.ac.uk

* Correspondence: pingwang@mail.ksu.edu.tw; Tel.: +886-6-205-0545

† This paper is an extended version of our paper published in 4rd IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability 2022 (IEEE ECBIOS2022), Tainan, Taiwan, 28–31 May 2022.

Abstract: With rapid advancements in in-vehicle network (IVN) technology, the demand for multiple advanced functions and networking in electric vehicles (EVs) has recently increased. To enable various intelligent functions, the electrical system of existing vehicles incorporates a controller area network (CAN) bus system that enables communication among electrical control units (ECUs). In practice, traditional network-based intrusion detection systems (NIDSs) cannot easily identify threats to the CAN bus system. Therefore, it is necessary to develop a new type of NIDS—namely, on-the-move Intrusion Detection System (OMIDS)—to categorise these threats. Accordingly, this paper proposes an intrusion detection model for IVNs, based on the VGG16 classifier deep learning model, to learn attack behaviour characteristics and classify threats. The experimental dataset was provided by the Hacking and Countermeasure Research Lab (HCRL) to validate classification performance for denial of service (DoS), fuzzy attacks, spoofing gear, and RPM in vehicle communications. The proposed classifier's performance was compared with that of the XBoost ensemble learning scheme to identify threats from in-vehicle networks. In particular, the test cases can detect anomalies in terms of accuracy, precision, recall, and F1-score to ensure detection accuracy and identify false alarm threats. The experimental results show that the classification accuracy of the dataset for HCRL Car-Hacking by the VGG16 and XBoost classifiers (n = 50) reached 97.8241% and 99.9995% for the 5-subcategory classification results on the testing data, respectively.

Keywords: in-vehicle network; car-hacking; HCRL dataset; VGG16; XGBoost

Citation: Lin, H.-C.; Wang, P.; Chao, K.-M.; Lin, W.-H.; Chen, J.-H. Using Deep Learning Networks to Identify Cyber Attacks on Intrusion Detection for In-Vehicle Networks. *Electronics* **2022**, *11*, 2180. https://doi.org/10.3390/electronics11142180

Academic Editors: Muhammad Salman Haleem, Liangxiu Han, Ernesto Iadanza and Baihua Li

Received: 17 June 2022
Accepted: 7 July 2022
Published: 12 July 2022

1. Introduction

With the rapid development of Internet of Vehicles (IoV) technology, electronic devices with in-vehicle networks (IVN) are becoming increasingly connected to the Internet to provide online communication and real-time updates on traffic. However, security and privacy issues, such as remote network intrusions into the controller area network (CAN), are critical concerns for IoV networks. In other words, while equipping existing vehicles with mobility services will create new business opportunities, it will also raise new cyber security threats for car manufacturers and consumers with the advent of the era of ubiquitous network connections and on-demand service such as over-the-air (OTA). Security vulnerabilities for OTA updates have been mitigated by process enhancements via user authentication with a system firmware examination.

According to a research report by Fortune Business Insights, the global electric vehicle (EVs) market size reached USD 246.70 billion in 2020. More than 10 million vehicles throughout the world's roads in 2020 were battery-powered electric models. The market is anticipated to grow from USD 287.36 billion in 2021 to USD 1318.22 billion in 2028, at

a CAGR of 24.3% in the 2021–2028 periods [1]. A series of security incidents targeting vehicular communications have occurred recently using denial of service (DoS), fuzzers, insertion, and spoofing gear/RPM attack techniques. Information security hazards include not only individual hackers highlighting their technical capabilities, but also group attacks aiming to extract financial gain. These attacks by hackers have concerns with car manufacturers. The hackers behind the aforementioned attacks demanded a large ransom from car manufacturers, and threatened to detonate money using implanted malicious codes.

Network communications are widely used in IVNs, which deploy CAN as a protocol for electronic control units (ECUs). ECUs can be used to control the transmission, engine, speed, airbags, powertrain, and many other subsystems of a vehicle. In practice, CAN suffers from multiple security issues, including the lack of data encryption and user authentication in inter- and intra-vehicle communications [2]. Generally, CAN buses are designed to guarantee reliable, rather than secure, communication in IVNs. For example, CAN uses the carrier sense multiple access with collision detection (CSMA/CD) protocol as a network communication, which allows nodes to capture unencrypted messages going through the network by a hacker. Furthermore, the ECUs in the CAN bus are vulnerable to cyberattacks because they lack security features such as user authentication and message encryption. Therefore, the improved CAN bus incorporates the network-based IDS on in-vehicle network system, namely, on-the-move intrusion detection system (OMIDS) into in-vehicle networks to help users detect abnormal connections in a timely manner [3].

The network-based IDS on an in-vehicle network (i.e., OMIDS) plays a role of data collector and data analyzer over the CAN bus that identifies security threats and attacks of in-vehicle network systems by using detection of anomalous CAN bus messages with protocol analyses [4]. Thus, distinct NIDSs on in-vehicle network systems are being developed for detecting threats of the CAN bus network attacks within in-vehicle communication protocol associated with potential attacks exploited, categorising the anomaly messages in securing in-vehicle networks and the related information systems on vehicles [2–7].

The concept of IDS deployed to the automotive system was first introduced by Hoppe et al. [5]; later, IDS deployment strategies were discussed by [6,7]. For NIDS to monitor and inspect the communication messages in the CAN network from different sources, it is recommended to be deployed to the central gateway [2]. As shown in Figure 1, the OMIDS was placed at a central gateway that monitors activities in the CAN network and identifies the attacks.

In the identification of possible attacks, OMIDS can be used to collect routing messages on the CAN Bus and identify attack categories for the IVN successfully. Therefore, a number of message records in the CAN bus must be collected, analysed, and classified in real time. Many classification approaches incorporate machine learning (ML) algorithms—such as the naïve Bayes classifier, decision tree, logistic regression (LR), support vector classifier (SVC), and deep convolutional neural networks [7–12]—to help managers precisely identify network attacks. In the future, OMIDS may encounter a more diverse range of attacks and extortions.

Most existing ML approaches for detecting cyber-attacks on OMIDS involve cyber-threat analyses that match routing information to potential attack profiles based on behavioural analysis techniques with packet collection, filtering, and feature comparisons. The routing information of CAN devices is used to classify possible attack categories, identify the real attack type, mark the compromised ECU, and initiate countermeasures.

Consequently, deep convolutional neural network techniques, such as recurrent neural networks (RNN) [8], long short-term memory (LSTM) [7], and convolutional neural networks (CNN) [9–12], have been adopted to help security managers detect complex threats from various sources. Typically, deep convolutional neural networks provide a new approach to improve threat recognition accuracy of network intrusion detection and reduce the false positive rate (FPR). However, a single base classifier does not sufficiently match the data distribution, either with a high bias (low-degree-of-freedom models) or high variance to be robust (high-degree-of-freedom models) [12]. Considering the increasing

security threats in the CAN bus systems, the challenges for automatic ML-based intrusion detection on the in-vehicle network systems are summarized as:

(i) The idea of machine learning-based IDS deployed to the CAN bus network was first introduced by Kang et al. in 2016 [13]. In this case, they used unsupervised pre-training of deep belief networks (DBN) model in detecting any deviations from normal frequencies of CAN message. Later, Taylor et al. used a support vector machine for binary classification to classify the CAN traffic flows [14]. Recently, Hossain et al. (2020) developed an LSTM to detect the threat predict using sequence data inputs and achieved an overall detection accuracy of 99.995% [7].

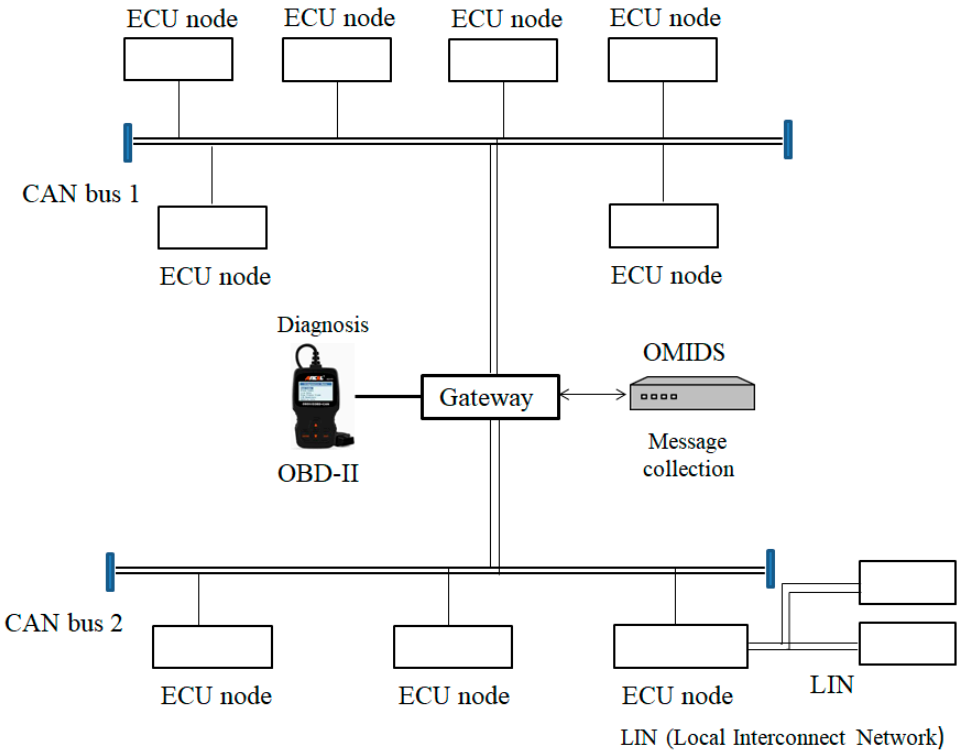


Figure 1. Deployment of the OMIDS within in-vehicle networking systems.

With the proliferation of attacks on an in-vehicle network, the traditional artificial neural network (ANN) failed to detect in some complex attack cases, such as fuzzy attack in [15]. In this case, Song et al. (2020) used the deep neural network models, deep convolutional neural network (DCNN), and LSTM to achieve high accuracy with low error rate for prediction results.

Theoretically, assembling multiple classifiers can reduce false positives and produce more accurate classification results than single classifiers [16]. For example, Rajadurai and Gandhi used a stacked ensemble model of different base classifiers to build a stronger learner and showed that the stacked ensemble model produced more accurate results than that of a single algorithm [17]. Therefore, ensemble learning-based techniques such as boosting [18], random forest [19], gradient boost DT (GBDT) [20], and XGBoost [21–24] have been developed to reduce the bias and variance.

(ii) In a supervised ML model, it requires complete labeled data in the training process. There are difficulties in predicting and generating attack behaviour in evaluating the CAN bus system [2]. Practically, existing signature-based approaches for the NIDS are based on behavioural features to categorize the threats [25]. Importantly, high accuracy of the OMIDS needs continuous updating for high-resolution feature set inputs extracted from attack scenarios of CAN bus, which is not a trivial task. Moreover, it cannot efficiently work if an unknown message is abnormal. Thus, the

development of an accurate and robust approach for automatic threat detection for in-vehicle network systems is still a challenge.

In this study, we used a CNN-based scheme called VGG16 [10] (a specific CNN model specialising in image recognition) to handle a diverse range of threat classification problems. The system validation incorporates the XBoost and naïve Bayes classifiers, decision tree, LR, and SVC to compare their performance.

In summary, the primary contributions of this study are as follows:

- For vehicle network security, this study addresses a diverse range of threat classification problems in intrusion detection systems using the VGG16 model to verify performance;
- In our experiment, the accuracy of the VGG16 model for intrusion detection is 100%/100% (Table 9) for binary classification on the training and testing data;
- As shown in Table 10, the proposed VGG16 approach provides higher prediction accuracy (97.9420%/97.8241%) for multiclass classification (five categories) than those of Naïve Bayes (91.0095%/91.0273%) and SVC classifier (91.0095%/91.4137%) on training and testing data;
- To compare the classification accuracy of competing approaches, including the XBoost classifier, Naïve Bayes classifier, DT, LR, and SVC, the test cases are capable of intrusion detection in terms of accuracy, precision, recall, and F1-score;
- In our experiment, the XBoost classifier's prediction error for intrusion detection decreased most steadily among the four models, followed by the VGG16 model, naive Bayes, decision tree, logistic regression, and SVC;
- The average accuracy of the classification of intrusion detection by the VGG16 and XBoost classifiers (n = 50) for the testing data was 97.8241% and 99.9995%, respectively.

The remainder of this paper is organised as follows: Section 2 reviews other relevant studies in the field. Section 3 introduces the proposed CNN-based model for in-vehicle network security analysis. Section 4 presents the performance analysis and its results. Finally, Section 5 provides concluding remarks.

2. Overview of Intrusion Detection for In-Vehicle Network

This section reviews approaches for addressing threats to in-vehicle networks, and introduces a CNN-based model, VGG16, and an ensemble method, XGBboost, for intrusion detection to categorise possible attacks.

2.1. Potential Vulnerabilities for In-Vehicle Network

The problem of identifying potential threats in an automotive communication system, specifically for the CAN bus protocol, is referred to as the security of the in-vehicle network (SOVN) problem. Typically, the SOVN problem involves a guarantee of data confidentiality, integrity, and availability that can never be breached within the ECUs or CAN bus in the security management of existing vehicles [3]. Solving the SOVN problem is of crucial concern in identifying possible attacks, given a constraint on both the quantity of routing CAN packets collected by OMIDS, and the computational time. Consequently, many detection methods for discovering true IVN attacks have recently been proposed.

Currently, the CAN bus fails to ensure all three essential security levels that would maintain necessary protection from diverse and complex threats. The CAN protocol does not have inherent cryptographic protections to ensure that only authorised parties have access to information. Consequently, it allows intruders to access sensitive user data and invade privacy. Furthermore, the CAN protocol does not feature a comprehensive integrity check, and fails to maintain the accuracy, completeness, and validity of data. Although the CAN bus has a CRC for the verification of integrity against transmission errors, it cannot prevent the injection of data by malicious parties. Given the nature of priority-based messaging in CAMA/CD, if a message with the highest priority is inserted, the network will be inaccessible by the lower-priority nodes, thus violating availability. Therefore, an OMIDS must be established for in-vehicle networks to detect possible threats under certain real-time constraints.

For instance, Miller and Valasek first demonstrated that they can hack into a Ford Escape and Toyota Prius, and control the brakes and steering through the use of physical access to the onboard diagnostics (OBD-II) computer port on each vehicle [26]. Later, Mahaffey and Rogers exploited the keyless flaws of Tesla Model S to remotely unlock the vehicle's doors, start the vehicle, and drive away by issuing a kill command to shut down the vehicle's systems, bringing it to a stop [27]. Recently, the Keen Security Lab of Tencent reported that attackers can exploit these flaws to take full control of a Tesla vehicle's infotainment system without any user interaction. In one experiment, they gained entrance from wireless Wi-Fi/Cellular, compromised of many in-vehicle systems, including firmware on IC, Gateway, and injected malicious messages into the CAN bus to perform a series of operations such as opening the car door, window, and trunk.

These security cases showed the existence of weak points in EVs that may affect the car manufacturers' reputations with substantial financial implications such as recalls. Safety and security are the highest short- and mid-term challenges in the automotive industry [28]. Therefore, extensive studies have been conducted to find possible solutions to the vulnerabilities of the CAN bus. Some of these studies have performed successful experimental attacks on cars [2,23–25,29,30]. Furthermore, researchers have proposed preventative methods for known attacks, including network segmentation, encryption, authentication, and intrusion detection systems (IDSs).

To address the lack of data encryption and user authentication of the CAN protocol in inter- and intra-vehicle communications, many studies on network detection have primarily focused on the use of digital signature mechanisms, such as symmetric secret key techniques, to defend against cyber threats [28]. In practice, digital signatures with pair-wise symmetric secret keys require a high computation time and communication overhead for in-vehicle networks. Because the bandwidth of CAN is limited to 500 kbps in the original design, this is not a practical approach for data encryption and user authentication. However, this approach may be realised for CAN-XL, which will support high transmission speeds of 10 Mbps in the future. However, most current approaches have been conducted with a low communication overhead on the intrusion detection system.

2.2. In-Vehicle Detection of Targeted CAN Bus Attacks

Modern vehicles' initiatives taken by the automotive manufacturers have increased the number of ECUs per vehicle. For example, there are nearly 70 ECUs deployed in the modern vehicle. With its increasing applications, CAN bus has become a standard choice for automobiles, as well as for other applications too such as EV batteries, planes, ships, machineries, and many more [31]. Practically, modern vehicles often use the CAN bus for communication among their components. From the intrusion reports [4–7,26,29,30], hackers exploited the vulnerabilities and intruded the in-vehicle network to compromise the targeted ECU of the vehicle and issue attack commands. As shown in Figure 2, an example of DoS attack injected high priority of CAN messages (0x000) in a short cycle from the compromised ECU node thru the use of OTA update and delayed the normal message communications.

Miller and Valasek [26] opened a new era of security analyses for intrusion detection in in-vehicle networks using open intrusion datasets published in 2013. In the experiments, we used well-known open datasets for the targeted intrusion detection of car hacking developed by the Hacking and Countermeasure Research Lab [32]. The datasets include normal flow as well as attack data, with the five major represented attack types being denial-of-service (DoS), fuzzers, insertion, spoofing gear/RPM, and hybrid attacks. Related details regarding attacks and detection methods are found in [3–6,29–32]. For instance, attackers often use fuzzy attacks to recognise the reaction of ECUs to certain data packets. In a fuzzy attack, spoofed random CAN ID and data packets are rapidly inserted into the CAN bus, causing network nodes to receive a large number of meaningless messages, which results in vehicle failure. Network attack types on the CAN bus are summarised in Table 1.

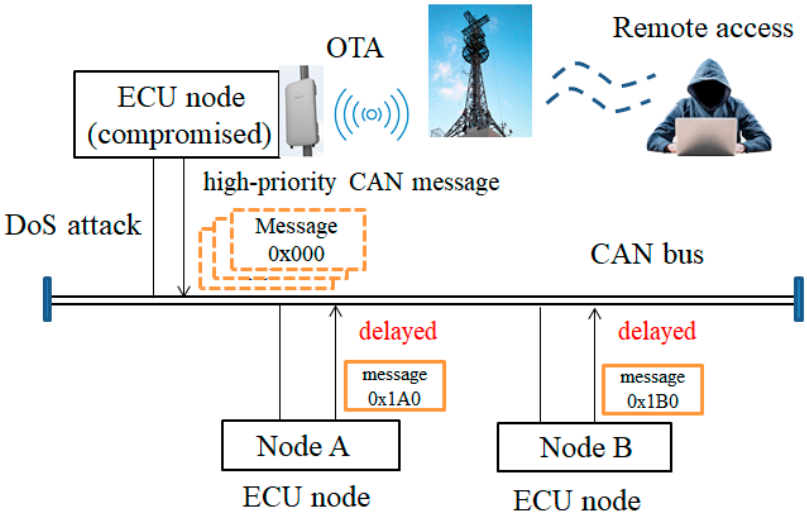


Figure 2. An example of DoS attack on in-vehicle networks.

Table 1. Common attacks for in-vehicle network.

	Features	Results/Countermeasures
DoS	Typically, DoS attacks are prepared by injecting high-priority CAN messages in a short cycle.	As the ECU that attempts to send a message with the most dominant CAN ID always wins the bus in the arbitration phase, other ECUs are prevented from transmitting their messages. The agent and SVM were used to improve the detection precision of intrusive attacks for network intrusion detection (NID).
Fuzzers	The fuzzy attack is similar to the DoS attack; however, the CAN ID and data values of messages are entirely random. Generally, fuzzy attacks are used to learn how ECUs react to certain packet types.	Spoofed random CAN ID and data packets are rapidly inserted into the CAN bus, causing failure. IDS detects the fuzzy attacks based on the differences from the insertion attack, because the random ID might not appear benign.
Insertion attacks	The method inserts packets to the CAN bus with an arbitrary ID and data frame. Unlike the fuzzy attack, insertion attack packets feature truly arbitrary IDs and data frame.	The use of insertion attacks might cause the vehicle to malfunction. Indeed, it is difficult to detect insertion attacks due the true arbitration of ID and data frame.
Spoofing gear/RPM	The spoofing attack enabled us to deceive the original ECU and change the RPM (revolution per minute) gauge and drive gear on the instrument panel.	The spoofing attack enabled us to deceive the original ECU and change the RPM gauge and the drive gear on the instrument panel.
Hybrid attacks	This method interleaves benign data with DoS attacks, fuzzers, spoofing gear/RPM, and insertion attacks.	The use of hybrid attacks may cause serious malfunctions in vehicles. In the identification of possible attacks, defenders used IDS on the CAN bus to collect sufficient routing information to identify attack categories successfully in the optimal time.

2.3. VGG16

VGG Net is a pre-trained CNN invented by Simonyan and Zisserman from the Visual Geometry Group (VGG) at the University of Oxford in 2014 [33]. This model obtained 92.7% test accuracy for ImageNet [34], winning it the ImageNet 2014 competition. To date, VGG16 is still considered an excellent vision model architecture, and has successfully been used in many real-world applications [10,35,36].

The standard VGG16 architecture consists of 13 convolutional layers, five max-pooling layers, three fully-connected layers (FCLs), and a softmax classifier. The number of filters in the original block was 64, though it was doubled in subsequent blocks until it reached 512 [10]. Thus, VGG16 offers a straightforward and simple architecture that sufficiently categorises cyber threats according to collected feature sets.

Inspired by [33], the VGG16 model parameters were used in the model training phase. In the developed model, the feature vectors were transformed into matrices, which formed VGG16 input images to accurately classify cyberattacks. Specifically, VGG16 uses a cascade of numerous layers of nonlinear processing units for feature extraction and transformation, and exhibits superior results in image recognition applications. The data were then clustered into categories according to classification weights. Consequently, the proposed model can minimise the classification error and maximise the generalisability of learning using convolutional features.

Classification is the most frequently encountered issue in ML. Most existing classification approaches for intrusion detection employ data mining algorithms to categorise cyber threats using feature sets by training a classifier. Classification problems based on convolutional features are illustrated as follows:

Consider a given training dataset $D(x_i, y_i)$, where x_i denotes the number of observations of a sample ($x_i \in R^N, i = 1, \ldots, n$), and y_i indicates the class to which the point x_i belongs $y_i, i = 1, \ldots, n$, y_i is assigned to each observation x_i. Each facial feature x_i is of a dimension that corresponds to the number of propositional variables:

$$f_\theta = \{(x_i, y_i)\} x_{i,} \in R^n, y_i \in \{0, 1, \ldots m\}\}_{i=1}^{n}, \quad (1)$$

Generally, a mapping function $f_\theta(x_i) = W \cdot x_i + b$ is defined for classification as follows:

$$\hat{y}^{(i)} \cong y^{(i)} = f_\theta(x_i) = s(W \cdot x_i + b), \quad (2)$$

where $\hat{y}^{(i)}$ is the final class of the output; $y^{(i)}$ represents the output result of the training process; W represents the convolution kernel of the feature matrix; b is the bias of the feature vector; f_θ represents a mapping function in the CNN; and s represents the softmax function.

2.4. XGBoost Classifier

Most previous studies [2,3,7–9] on intrusion detection in in-vehicle networks focus on detecting abnormal CAN messages using a base classifier such as DT, LR, SVC, CNN, or LSTM, as shown in Figure 3.

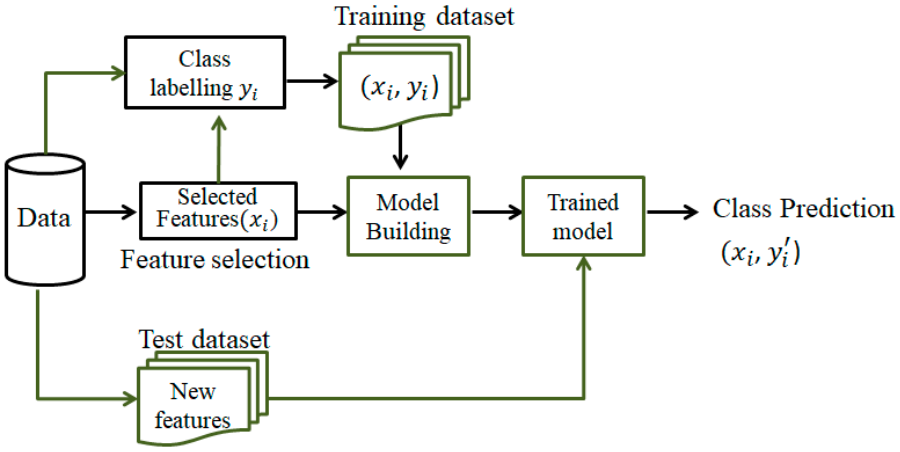


Figure 3. General architecture of a base classifier.

As shown in Figure 3, a base (individual) classifier in ML models use the selected feature vectors from the training instances to train the model, determine the hyperparameters of the trained model, and predict the possible categories of test data. In the training

process, there are important topics for the ML model: (i) feature selection. Feature selection is the process of reducing the number of input variables to develop a predictive model using a selected feature set to improve the performance of the model by reducing the computational costs of modelling [37]. (ii) hyperparameter tuning. Essentially, the prediction performance of the machine learning network model is influenced quite heavily by the choice of hyperparameters, hence it can corporate the grid search optimisation process to improve the searching efficiency in model development [38].

As mentioned above, ensemble learning approach allows the production of better predictive performance compared to a single model [16]. Ensemble learning algorithms, such as extreme gradient boosting (XGBoost), can overcome overfitting in threat data. In particular, XGBoost energises machine learning model performance and computational speed, where decision trees are built in parallel instead of sequentially, as in GBDT.

To overcome overfitting in threat data and increase detection accuracy of normal and intrusive patterns, this paper proposes an improved ensemble-based learning algorithm associated with an XGBoost classifier to identify features from qualified threats.

XGBoost is an efficient implementation of the gradient boosting machine learning algorithm that employs stochastic gradient or tree boosting to create a powerful machine learning technique that performs well on a wide range of challenging problems [21]. It has been shown to provide state-of-the-art results on several standard classification benchmarks. Furthermore, XGBoost is a scalable and highly accurate implementation of gradient boosting that limits the computing power of boosted tree algorithms. In other words, XGBoost provides parallel tree boosting, and is the leading machine-learning library for regression, classification, and ranking problems (Figure 4).

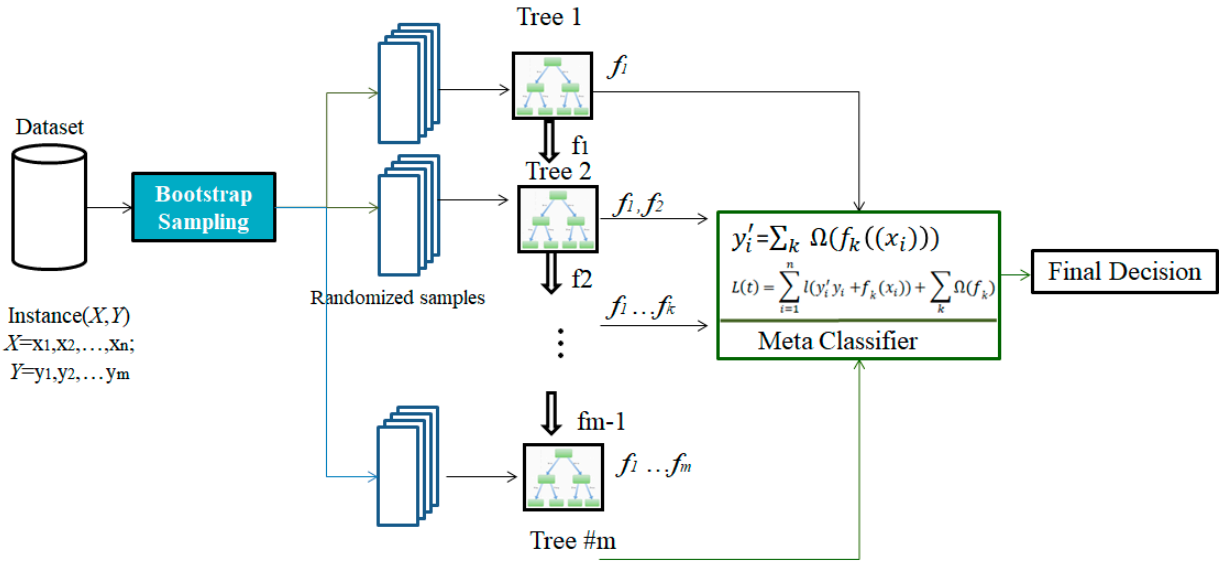


Figure 4. Operational flow of the XGBoost classifier.

As shown in Figure 2, XGBoost is an ensemble of decision tree algorithm, where new trees fix the errors present in trees that were already part of the model. New trees are generated until no further improvements can be made to the model. XGBoost provides a highly efficient implementation of the stochastic gradient boosting algorithm, as well as access to a suite of model hyperparameters designed to provide control over the model-training process [21]. The most significant factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single ML machine, and scales to billions of examples in distributed and memory-limited settings.

XGBoost Features

XGBoost improves upon the gradient boosting algorithm, where the Newton–Raphson–Gradient method is used to explore the solution of the loss function by expanding its Taylor series to the second order, as well as adding a regularisation term. The objective function during training consists of two parts: the loss of the gradient boosting algorithm and the regularisation term.

The loss function of the XGBoost model is defined as [21,23]:

$$L(t) = \sum_{i=1}^{n} l(y_i' y_i + f_k(x_i)) + \sum_{k} \Omega(f_k), \tag{3}$$

where l is the loss function, f_k is the kth tree output, and Ω is a normal term. Assuming that $L(t)$ is a convex function, y_i' represents the predicted value for the training sample at time $t-1$, and y_i denotes the true value of the training sample at time t. One major approach to ensure fast calculation is the approximation of the Taylor expansion as:

$$L(t) \approx \sum_{i=1}^{n} l(y_i' y_i) + g_i f_k(x_i) + \frac{1}{2} h_i {f_k}^2 + \sum_{k} \Omega(f_k), \tag{4}$$

where g_i and h_i are the loss function's first and second derivatives, respectively. The normal term defines the model's complexity, and can be simplified as

$$\Omega(f_k) = \gamma \mathrm{T} + \frac{1}{2} \lambda ||w||^2, \tag{5}$$

where γ and λ are the given parameter values, w is the vector formed by the values of all leaf nodes of the decision tree, and T is the number of leaf nodes. XGBoost was developed into an optimised distributed function library for the gradient boosting model, with the purpose of achieving high efficiency, flexibility, and portability.

3. An Analysis Model for Intrusion Detection of an In-Vehicle Network

The proposed network intrusion detection model combines the CNN model with an ensemble learning XGBoost algorithm to maintain high precision in predicting the stability of network intrusion detection after the collection of suspicious network flows. The overall structure of the model is exhibited in Figure 5, which illustrates the three sub-phases in the behaviour classification process: (1) data pre-processing, (2) model training, and (3) model validation.

Step 1. Data Preprocessing

First, the training sample data were obtained from two data sources: the common HCRL intrusion detection archive [32], and the HCRL Car-Hacking dataset, the latter of which were used in several other CAN IDS case studies [39,40]. The HCRL dataset contains 30–40 min of CAN traffic, with approximately 4 million total messages. For our approach, we only evaluated spoofing attacks on the driving gear and RPM gauge from this set. In these attacks, only the messages for a single ID out of approximately 25 were attacked in each case. Therefore, only messages with these IDs were evaluated.

Step 1.1. Experiment Data Sources

In the intrusion detection experiment, the overall HRCL dataset was selected as a comprehensive dataset to examine the performance of the developed XGBoost classifier. The HRCL dataset was compiled by Culture Makers and the Korea Internet and Security Agency for car security to include synthetic contemporary attack behaviours in real-world in-vehicle network traffic.

To extensively examine the developed model's performance, we reconstructed the experimental dataset to generate five major attack types by comprising a CAN-intrusion dataset and a CAN-hacking dataset. To simplify the amount of training data and improve efficiency of the training process, redundant normal messages were removed and injected messages were extracted.

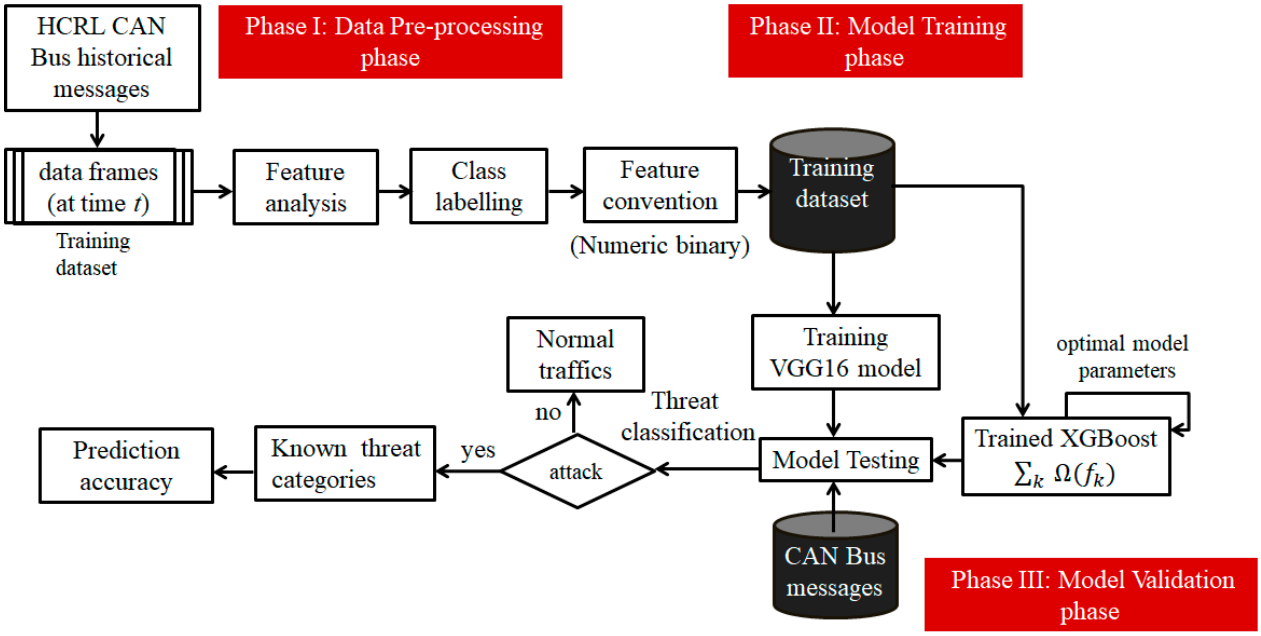


Figure 5. Operation flow of network intrusion detection for in-vehicle networks.

Step 1.2. Normalisation

All fields in the HRCL dataset were regarded as symbolic features. First, we performed symbol conversion of the network packets. Then, the attack categories of Flag were converted to a numerical format (1–5) according to different attack types, where normal traffic was assigned to '0' through a one-hot encoding process.

Step 2. Model Training Phase

In this step, the experimental data were divided into a training set and a testing set. The training set contained 175,341 records (68.05%), whereas the testing set contained 82,332 records (31.95%). Furthermore, the training set contained 119,341 (68.06%) and 56,000 (31.95%) attack and normal files, respectively, whereas the testing set had 45,332 (55.06%) attack files and 37,000 (44.94%) normal files.

For the model training phase, the VGG16 model was used to accurately categorise cyber threats. In the experiment, the revised VGG16 was trained to detect network intrusion based on the behavioural patterns of collected samples. The softmax function was used to evaluate the model, and a cross-entropy function was used to adjust the learning rate to minimise the classification error and increase the speed of the training process.

Step 3. Model Validation Phase

In the model validation phase, the trained model's performance was assessed and compared using the XGBoost classifier with Equations (3)–(5). After training the base classifiers, the results were aggregated from randomly selected sub-datasets of the training data; finally, a meta-classifier was trained to perform threat classification of the samples.

In practice, the DLN training process and classification performance of the model are significantly influenced by the choice of hyperparameters. Therefore, we used GridSearchCV to set the model parameters for multi-classification on XGBoost. The learning results were used as a basis for the parameters in the validation phase, including weight matrix, batch size, epochs, and classification accuracy.

4. Experimental Results

All experiments were conducted in Python using the ML library of the scikit-learn package, which is an open-source library for classification algorithms. The software, presented in Table 2, was run on an Intel Core i3-4160 dual core CPU clocked at 3.0 Ghz and 8 GB of DDR3 RAM. The operating system was Ubuntu Desktop 20.04.3 LTS, and the database platform was MongoDB 5.0.3.

Table 2. Experimental environment.

Numerical and Machine Learning Library	
Python 3.8.10	Tensor flow 2.1.0 scikit-learn Numpy 1.21.5 scipy 1.0.1 Pandas 1.2.0

Step 1. Data Pre-Processing Phase

Step 1.1. Experiment Data Sources

The experimental dataset for HCRL Car-Hacking comprises four major types of network attacks: DoS, fuzzy, spoofing the gear gauze (gear spoofing), and spoofing the RPM gauze (RPM spoofing), as shown in Table 3.

Table 3. Overview of the experiment HCRLCar-Hacking dataset.

Attack Type	# of Injected Messages	# of Normal Messages	Total
DoS Attack	587,521 (16.03%)	3,078,250 (83.97%)	3,665,771 (100.0%)
Fuzzy Attack	491,847 (12.81%)	3,347,013 (87.19%)	3,838,860 (100.0%)
Gear Spoofing	597,252 (13.44%)	3,845,890 (86.56%)	4,443,142 (100.0%)
RPM Spoofing	654,897 (14.17%)	3,966,805 (85.83%)	4,621,702 (100.0%)

We selected the HRCL dataset because it offers three advantages over other benchmark datasets. First, it contains up-to-date behavioural features with existing attack sequences of CAN messages. Furthermore, it includes a set of features from the CAN IDs (header of packets) and DATA to reflect the network packets efficiently. In addition, it contains many complex features that the model can learn to discriminate more accurately, as shown in Table 4, which lists the numbers of normal and malicious messages in the experimental datasets. Referring to HRCL attack scenarios, we reconstructed and aggregated four major attack datasets: DoS attack, fuzzy attack, spoofing the drive gear, and spoofing the RPM gauge. Each dataset was generated by simulating ECUs while injecting fabricated CAN messages in a controlled environment. These four basic attack messages were then mixed and grouped for our experimental dataset, which was subsequently divided into training and testing sets.

Table 4. Details of experiment messages.

Attack Type	No. of Record	Message Details
Normal	14,237,958(85.93%)	Normal CAN messages
DoS	587,521(3.55%)	Inject the message with CAN ID of '0x000'
Fuzzy	491,847(2.97%)	Randomly inject deceptive messages with CAN ID and DATA
Gear Spoofing	597,252(3.6%)	Inject the messages of imitating nodes with Arbitrary ID = '0x43f'
RPM Spoofing	654,897(3.95%)	Inject the messages of imitating nodes with Arbitrary ID = '0x316'
Total	16,569,475(100%)	

Then, four CSV files of HCRL Car-Hacking were merged into 16,569,475 messages with a balanced sampling policy to enhance the precision of the multiclass classification model. In other words, the four types of attacks are equivalent to distributing with similar probabilities in the experiment dataset. Relevant data are presented in Table 4.

The CAN message data attributes in Table 4 include the timestamp, CAN ID, DLC, DATA [0], DATA [1], DATA [2], DATA [3], DATA [4], DATA [5], DATA [6], DATA [7], and

flag fields which are summarized as Table 5. To perform multi-label classification, we specified the class label for each CAN message in the model training data, as shown in Figure 6. Therefore, a set of reduced features with 10 feature sets, including items 2–5, was used for the threat classification of in-vehicle networks:

Timestamp: recorded time (s)
CAN ID: identifier of CAN message in HEX (ex. 043f)
DLC: number of data bytes, from 0 to 8
DATA [0~7]: data value (byte)
Flag: T or R, T represents injected special attack messages while R represents normal messages

Table 5. Data attributes for CAN message.

Field	Description of CAN Message
Timestamp	recorded time (s)
CAN ID	identifier of CAN message in HEX (ex. 043f)
DLC	number of data bytes, from 0 to 8
DATA [0~7]	data value (byte)
Flag	T or R, where 'T' represents injected special attack messages while 'R' represents normal messages

The training set contained 13,255,580 records (80.0%), whereas the testing set contained 3,313,598 records (20.0%), including malicious and normal files. The training set had 11,390,366 (85.93%) and 1,865,214 (14.07%) normal and intrusion attack files, respectively. The testing set had 2,847,375 (85.93%) and 466,223 (14.07%) normal and intrusion attack files, respectively.

	Timestamp	CAN ID	DLC	DATA(0)	DATA(1)	DATA(2)	DATA(3)	DATA(4)	DATA(5)	DATA(6)	DATA(7)	Flag	Type	label
0	1.478198e+09	0130	8.0	18	80	00	ff	1b	80	0a	e4	R	Normal	0
1	1.478198e+09	0131	8.0	00	80	00	00	4c	7f	0a	9a	R	Normal	0
2	1.478198e+09	0140	8.0	00	00	00	00	08	25	2a	47	R	Normal	0
3	1.478198e+09	0370	8.0	00	20	00	00	00	00	00	00	R	Normal	0
4	1.478198e+09	043f	8.0	00	40	60	ff	7e	cb	08	00	R	Normal	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4659648	1.478193e+09	316	8.0	45	29	24	ff	29	24	0	ff	T	RPM	5
4659649	1.478193e+09	316	8.0	45	29	24	ff	29	24	0	ff	T	RPM	5
4659650	1.478193e+09	316	8.0	45	29	24	ff	29	24	0	ff	T	RPM	5
4659651	1.478193e+09	316	8.0	45	29	24	ff	29	24	0	ff	T	RPM	5
4659652	1.478193e+09	316	8.0	45	29	24	ff	29	24	0	ff	T	RPM	5

Figure 6. Samples of training data set with class labels.

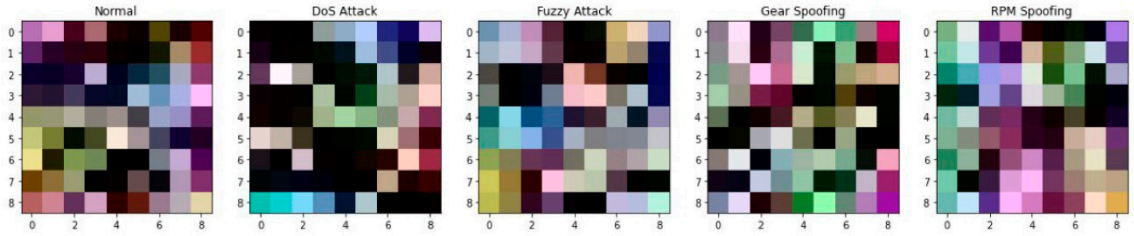
Step 1.2. Normalisation

All fields in the HRCL dataset were regarded as symbolic features. First, we performed symbol conversion on the network packets. Then, the attack categories of Flag were converted to a numerical format (1–5) according to different attack types, and normal traffic was assigned to '0' through a one-hot encoding process.

Step 1.3. Feature Conversion

In the following, the feature vectors were transformed into feature images (i.e., image matrix transformation), which formed the input images of the VGG16 model to accurately categorise cyber threats in in-vehicle networks. To recognise the feature differences between malicious attacks and normal applications in binary format, we used a feature matrix to represent the behavioural features of traffic connections using the quantile normalisation formula. In statistics, the quantile normalisation process is a technique for equalising the statistical properties of two distributions. Thus, quantile normalisation was implemented to form a feature map for model training. As shown in Figure 7, the feature maps generated

by the normal application and malware from the HRCL dataset are significantly different. Therefore, they can be used for image classification.



Visualisation of feature matrix for normal, DoS, fuzzy, gear spoofing, and RPM spoofing data

Figure 7. Feature map generated by the feature similarity matrix.

Step 2. Model Training and Optimization Phase

Inspired by [33], the VGG16 model parameters were used in the model training phase, as listed in Table 6. VGG16 is a CNN architecture used to win the ImageNet 2014 competition. To date, it is considered an excellent vision model architecture.

Table 6. Architecture of the revised VGG16 model.

Layer	No. of Filter	Filter Size	Feature Map Generated
Convolution layer C_1	64	$3 \times 3 \times 3$	$150 \times 150 \times 64$
Pooling layer P_1	1	2×2	$75 \times 75 \times 64$
Convolution layer C_2	128	$3 \times 3 \times 64$	$150 \times 150 \times 128$
Pooling layer P_2	1	2×2	$37 \times 37 \times 128$
Convolution layer C_3	256	$3 \times 3 \times 128$	$37 \times 37 \times 256$
Pooling layer P_3	1	2×2	$18 \times 18 \times 256$
Convolution layer C_4	512	$3 \times 3 \times 256$	$18 \times 18 \times 512$
Pooling layer P_4	1	2×2	$9 \times 9 \times 512$
Convolution layer C_5	512	$3 \times 3 \times 512$	$9 \times 9 \times 512$
Pooling layer P_6 (Max)	1	2×2	$4 \times 4 \times 512$
Pooling layer P_7 (Avg)	1	2×2	512
Dense (256)	-	-	256
Dropout (rate = 0.5)	-	-	256
Classification (Softmax)	-	-	2 for binary classification 5 for multi-classification

VGG16 comprises 13 convolutional layers, five max-pooling layers, and three fully-connected layers. The number of filters in the first block is 64, and is doubled in subsequent blocks until it reaches 512. As listed in Table 6, each VGG block consists of a sequence of convolutional layers, which are followed by a max-pooling layer. The same kernel size (3×3) was applied to all convolutional layers.

In the experiment, the input size is a $150 \times 150 \times 3$ image converted from a set of network features, batch_size = 32, epoch = 20, strides = (1, 1), activation function= 'tanh', loss = 'categorical_crossentropy', optimiser = 'dadelta'. The learning results were used as a basis for the validation parameters, including weight matrix, batch size, epochs, and classification accuracy. The experimental results of model training using VGG-16 are shown in Figure 8 and Table 7.

Step 3. Model Validation Phase

To validate the prediction accuracy of the developed model, we applied the XGBoost classifier to ensure that the prediction accuracy of a meta-classifier, such as XGBoost, is superior to that of a base classifier. In practice, it is necessary to first determine the number of trees (component classifiers) to be generated in the XGBoost classifier. Very few practical cases involve more than 300 trees, and such cases may significantly increase

the computation time for learning [41]. In this study, GridSearchCV with a 10-fold cross-validation scheme was used to set the model parameters of multi-classification on XGBoost. The search parameters for the XGBoost classifier are listed in Table 8 and Figure 9.

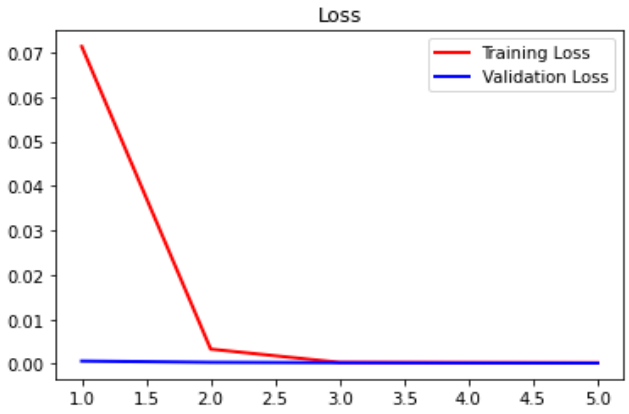


Figure 8. Training and validation loss of VGG16 for binary classification.

Table 7. Binary classification accuracy of VGG16 model.

Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Training Time (s)
100.0	100.0	100.0	100.0	389 sec

Table 8. Initial parameter search for XGBoost classifier using GridSearchCV.

Model \ Parameter	*n*-Estimators	Max-Depth	Learning Rate	Optimizer
XGBoost	[50, 100, 150, 200, 250, 300]	[4, 5, 6, 7, 8]	0.0001	Negative Log Likelihood Loss

```
# XGBoost with Gridsearchcv
cv = 10
xgb = XGBClassifier(objective="multi:softprob",
                    eval_metric='mlogloss',
                    use_label_encoder=False)
parameters = {
    "max_depth": [i for i in range(2, 8)],
    "n_estimators": [50, 100, 150, 200, 250, 300],
    "learning_rate": [0.0001, 0.001, 0.01, 0.1, 1.0],
}

xgbclf = GridSearchCV(estimator=xgb,
                      param_grid=parameters,
                      cv=cv,
                      n_jobs=2)

xgbclf.fit(X_train, y_train)
```

Figure 9. Searching for hyperparameters using 10-fold cross-validation with GridSearchCV.

In this study, experiments to determine the appropriate number of trees on the XGBoost classifier were conducted by examining overall accuracy with different numbers of trees ranging from 50 to 300. The low classification error of the XGBoost classifier shown in Figure 10 was obtained with data approximately 50 trees.

From Table 9, it is seen that there is almost the same performance (i.e., accuracy precision, recall, F1 Score, and ROC AUC) of multi-classification in six experiments (n = 50, 100, 150, 200, 250, 300). Notably, there is a low loss of cost function when training with the training set for 50 trees (n = 50), compared to n =100, 150, 200, 250, 300. Accordingly, the number of trees for the XGBoost classifier was set to 50, to produce the optimal benefit in prediction performance from learning additional trees. The performance of the XGBoost

classifier was then evaluated using Equations (3)–(5) by training component classifiers and aggregating their results by randomly selecting sub-datasets of the training data. Finally, a meta-classifier was trained with a majority voting approach to perform threat classification of the samples.

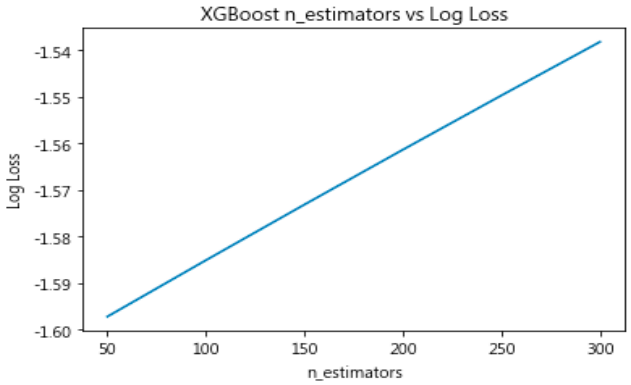


Figure 10. Binary classification error with a given number of trees.

Table 9. Multi-classification performance with the number of trees selected for the XGBoost classifier.

	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	ROC AUC (%)
n = 50	99.7880	99.9480	98.5739	99.2347	99.7084
n = 100	99.7881	99.9480	98.5745	99.2350	99.7084
n = 150	99.7881	99.9480	98.5745	99.2350	99.7084
n = 200	99.6991	99.9284	97.9733	98.8968	99.9450
n = 250	99.7888	99.9482	98.5790	99.2375	99.7084
n = 300	99.7888	99.9482	98.5790	99.2375	99.7084

The average accuracy for both VGG16 and XGBoost classifiers (n = 50) was approximately 100.0% (Table 10) for the binary classification results of the training and testing data. The accuracy results (in %) associated with the optimal C and γ values were obtained through a cross-validation scheme for the SVC. The binary classification accuracy rates for the training and testing datasets were approximately 100.0% and 100.0% (C = 1000, γ = 0.1), respectively.

Table 10. Binary classification accuracy when 10 features were used.

	Training	Testing
VGG16	100.0%	100.0%
Naïve Bayes	100.0%	100.0%
CART	100.0%	100.0%
Logistic Regression	100.0%	100.0%
SVC	100.0%	100.0%
XGBoost classifier	100.0%	100.0%

The optimal parameters C = 1000 and γ = 0.1, for SVC were examined using Python GridSearchCVparam = {'C_range':(0.1, 10, 100, 1000), 'Gamma_range':(0.1, 10, 100, 1000)}, and the fitting error with mean_squared_error was analysed for different values of C and γ. After analysing the experimental results, C = 1000 and γ = 0.1 for SVC were selected. Table 10 shows the same binary classification performance (i.e., accuracy = 100%, recall = 100%, precision = 100%, F1 Score = 100%, and ROC AUC = 100%) for the VGG16 model and the four base classifiers, compared to the XGBoost classifier (n = 50).

Similarly, using the VGG16 and XGBoost classifiers (n = 50) on the testing data, the multiclass classification accuracy levels for the five subcategories decreased to 97.8241% and 99.9995%, respectively, as listed in Table 11.

Table 11. Multiclass classification accuracy when 10 features were considered.

	Training	Testing
VGG16	97.9420%	97.8241%
Naïve Bayes	91.0095%	91.0273%
CART	99.3259%	99.3170%
Logistic Regression	99.3170%	98.9261%
SVC	91.0095%	91.4137%
XGBoost classifier	99.9997%	99.9995%

In addition, the multiclass accuracy levels (%) of naïve Bayes decreased to 91.01% and 91.03% for the testing data. Similarly, the multiclass accuracies (%) of CART decreased to 99.32% on the testing data. Obviously, there is better multi-classification performance for the XGBoost model (n = 50) compared to VGG16 and the four base classifiers (i.e., naïve Bayes, CART, logistic regression, and SVC).

5. Conclusions

This paper presents an intrusion detection model that incorporates an XGBoost classifier with an ensemble of a decision tree algorithm to enhance the precision of a multiclass classification model for in-vehicle network security. Moreover, the proposed approach minimises classification errors using a balanced class with a set of reduced features to accelerate intrusion detection. Overall, the results indicate that the precision of the proposed model for the XGBoost classifier in the intrusion detection analysis of in-vehicle networks is higher than that for the VGG16 model, and those of the four base classifiers, on the HCRL Car-Hacking dataset from Table 11.

Author Contributions: Conceptualization, P.W.; methodology H.-C.L. and K.-M.C.; resources, P.W.; formal analysis, H.-C.L.; data curation, H.-C.L. and J.-H.C.; writing—original draft, H.-C.L. and W.-H.L.; writing— review and editing, P.W. and K.-M.C.; software, H.-C.L. and J.-H.C.; validation, H.-C.L. and W.-H.L.; visualization, H.-C.L. and J.-H.C.; project administration, P.W.; funding acquisition, P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology of Taiwan under Grant No. MOST 110-2410-H-168-003 and the green energy technology research center on the featured areas research center program within the framework of the higher education sprout project from Ministry of Education (MOE) of Taiwan under Grant No. MOE 2000-109CC5-001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fortune, the global Electric Vehicle Market Is Anticipated to Grow from $287.36 Billion in 2021 to $1318.22 Billion in 2028 at a CAGR of 24.3% in Forecast Period, Electric Vehicle. Available online: https://www.fortunebusinessinsights.com/industry-reports/electric-vehicle-market-101678 (accessed on 15 January 2021).
2. Lokman, S.F.; Othman, A.T.; Abu-Bakar, M.-H. Intrusion Detection System for Automotive Controller Area Network (can) Bus System: A Review. *EURASIP J. Wirel. Commun. Netw.* **2019**, *1*, 184. [CrossRef]
3. Han, M.L.; Kwak, B.I.; Kim, H.K. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Veh. Commun.* **2018**, *14*, 52–63. [CrossRef]
4. Hoppe, T.; Kiltz, S.; Dittmann, J. Applying intrusion detection to automotive it-early insights and remaining challenges. *J. Inform. Assur. Secur.* **2009**, *4*, 226–235.
5. Apvrille, L.; El Khayari, R.; Henniger, O.; Roudier, Y.; Schweppe, H.; Seudié, H.; Weyl, B.; Wolf, M. Secure automotive on-board electronics network architecture. In Proceedings of the FISITA World Automotive Congress, Budapest, Hungary, 30 May–4 June 2010; Volume 8.

6. Studnia, I.; Alata, E.; Nicomette, V.; Kaâniche, M.; Laarouchi, Y. A languagebased intrusion detection approach for automotive embedded networks. *Int. J. Embed. Syst.* **2018**, *10*, 1. [CrossRef]
7. Hossain, M.D.; Inoue, H.; Ochiai, H.; Fall, D.; Kadobayashi, Y. LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications. *IEEE Access* **2020**, *8*, 185489–185502. [CrossRef]
8. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef]
9. Rehman, A.; Ur Rehman, S.; Khan, M.; Alazab, M.; Thippa, R.G. Canintelliids: Detecting In-vehicle Intrusion Attacks on A Controller Area Network Using CNN and Attention-based GRU. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 1456–1466.
10. Qassim, H.; Verma, A.; Feinzimer, D. Compressed Residual-VGG16 CNN Model for Big Data Places Image Recognition. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 169–175.
11. Zhang, H.; Gu, M.; Jiang, X.D.; Thompson, J.; Cai, H.; Paesani, S.; Santagati, R.; Laing, A.; Zhang, Y.; Yung, M.H.; et al. An Optical Neural Chip for Implementing Complex-valued Neural Network. *Nat. Commun.* **2021**, *12*, 457. [CrossRef]
12. Zhu, H.H.; Zou, J.; Zhang, H.; Shi, Y.Z.; Luo, S.B.; Wang, N.; Cai, H.; Wan, L.X.; Wang, B.; Jiang, X.D.; et al. Space-efficient Optical Computing with an Integrated Chip Diffractive Neural Network. *Nat. Commun.* **2022**, *13*, 1044. [CrossRef]
13. Kang, M.J.; Kang, J.W. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE* **2016**, *11*, e0155781. [CrossRef]
14. Taylor, A.; Japkowicz, N.; Leblanc, S. Frequency-based anomaly detection for the automotive CAN bus. In Proceedings of the 2015 World Congress on Industrial Control Systems Security (WCICSS), London, UK, 14–16 December 2015; pp. 45–49.
15. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]
16. Mahfouz, A.; Abuhussein, A.; Venugopal, D.; Shiva, S. Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset. *Future Internet* **2020**, *12*, 180. [CrossRef]
17. Rajadurai, H.; Gandhi, U.D. A stacked ensemble learning model for intrusion detection in wireless network. *Neural Comput. Applic* **2020**, 1–9. [CrossRef]
18. Rocca, J. Ensemble Methods: Bagging, Boosting and Stacking, towards Data Science, 23 April 2019. Available online: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205 (accessed on 22 February 2021).
19. Andy, L.; Matthew, W. Classification and Regression by Random Forest. *R. News* **2002**, *2*, 18.
20. Gradient Boosting. Available online: https://wikipedia.org/wiki/Gradient%20boosting (accessed on 22 February 2021).
21. Dhaliwal, S.S.; Al Nahid, A.; Abba, R. Effective Intrusion Detection System Using XGBoost. *Information* **2018**, *9*, 149. [CrossRef]
22. Ha, N. TreeBoosting-03: Why does XGBoost Win Every Machine Learning Competition? *Data Science Blog.* Available online: https://datasciblog.github.io/2020/02/26/tree-boosting-03/ (accessed on 22 February 2021).
23. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
24. Omer, S.; Lior, R. Approximating XGBoost with an interpretable decision tree. *Inf. Sci.* **2021**, *572*, 522–542.
25. Singh, J.; Nene, M.J. A Survey on Machine Learning Techniques for Intrusion Detection Systems. *Int. J. Adv. Res. Comput. Commun. Eng.* **2013**, *2*, 4349–4355.
26. Reuters, Hackers to Release Techniques for Attacking Toyota Prius, Ford Escape. 29 July 2013. Available online: https://www.wheels.ca/news/hackers-to-release-techniques-for-attacking-toyota-prius-ford-escape (accessed on 22 February 2021).
27. Walker, M. Security Experts Reveal How a Tesla Model S Was Hacked, the Hollywood Report, 7 August 2015. Available online: https://www.hollywoodreporter.com/news/general-news/security-experts-reveal-how-a-814062/ (accessed on 15 January 2021).
28. Le, V.H.; Hartog, J.; Zannone, N. Security and Privacy for Innovative Automotive Applications: A Survey. *Comput. Commun.* **2018**, *132*, 17–41. [CrossRef]
29. Islam, R.; Refat, R.U.D.; Yerram, S.M.; Malik, H. Graph-based Intrusion Detection System for Controller Area Networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *23*, 21664787. [CrossRef]
30. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
31. Precedence Research, Automotive Communication Technology Market Size, Report 2021–2030. Available online: https://www.precedenceresearch.com/automotive-communication-technology-market (accessed on 3 July 2022).
32. HCRL Dataset. Available online: https://goo.gl/WiVeFj (accessed on 22 February 2021).
33. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
34. Hassan, M.U. VGG16–Convolutional Network for Classification and Detection, 20 November 2018. Available online: https://neurohive.io/en/popular-networks/vgg16/ (accessed on 30 April 2019).
35. Lee, H.; Whang, M. Heart Rate Estimated from Body Movements at Six Degrees of Freedom by Convolutional Neural Networks. *Sensors* **2018**, *18*, 1392. [CrossRef] [PubMed]
36. Gopalakrishnan, K.; Khaitan, S.; Agrawal, A. Deep Convolutional Neural Networks with Transfer Learning for Computer Vision-based Data-driven Pavement Distress Detection. *Constr. Build. Mater.* **2017**, *157*, 322–330. [CrossRef]

37. Lin, H.C.; Wang, P.; Chao, K.M.; Lin, W.H.; Yang, Z.Y. Ensemble learning for threat classification in network intrusion detection on a renewable energy security monitoring system. *Appl. Sci.* **2021**, *11*, 11283. [CrossRef]
38. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40.
39. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN Based Intrusion Detection System for In-vehicle Network. In Proceedings of the 2018 16th IEEE Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018.
40. Verma, M.E.; Iannacone, M.D.; Bridges, R.A.; Hollifield, S.C.; Moriano, P.; Kay, B.; Combs, F.L. Addressing the Lack of Comparability & Testing in CAN Intrusion Detection Research: A Comprehensive Guide to CAN IDS Data & Introduction of the Road Dataset. *arXiv* **2012**, arXiv:2012.14600.
41. Ramon, J. Comment on: How to Determine the Number of Trees to Be Generated in Random Forest Algorithm. Available online: https://www.researchgate.net/post/How_to_determine_the_number_of_trees_to_be_generated_in_Random_Forest_algorithm (accessed on 12 September 2021).