Article

A Kohonen SOM Architecture for Intrusion Detection on In-Vehicle Communication Networks

Vita Santa Barletta [1], Danilo Caivano [1], Antonella Nannavecchia [2],* and Michele Scalera [1]

[1] Department of Informatics, University of Bari Aldo Moro, Via E. Orabona 4, Bari 70125, Italy; vita.barletta@uniba.it (V.S.B.); danilo.caivano@uniba.it (D.C.); michele.scalera@uniba.it (M.S.)

[2] Department of Economics and Management, University LUM Jean Monnet, SS 100 km 18, Casamassima (BA) 70010, Italy

* Correspondence: nannavecchia@lum.it

Abstract: The diffusion of connected devices in modern vehicles involves a lack in security of the in-vehicle communication networks such as the controller area network (CAN) bus. The CAN bus protocol does not provide security systems to counter cyber and physical attacks. Thus, an intrusion-detection system to identify attacks and anomalies on the CAN bus is desirable. In the present work, we propose a distance-based intrusion-detection network aimed at identifying attack messages injected on a CAN bus using a Kohonen self-organizing map (SOM) network. It is a power classifier that can be trained both as supervised and unsupervised learning. SOM found broad application in security issues, but was never performed on in-vehicle communication networks. We performed two approaches, first using a supervised X–Y fused Kohonen network (XYF) and then combining the XYF network with a K-means clustering algorithm (XYF–K) in order to improve the efficiency of the network. The models were tested on an open source dataset concerning data messages sent on a CAN bus 2.0B and containing large traffic volume with a low number of features and more than 2000 different attack types, sent totally at random. Despite the complex structure of the CAN bus dataset, the proposed architectures showed a high performance in the accuracy of the detection of attack messages.

Keywords: intrusion detection system; supervised learning; Kohonen self-organizing maps; CAN bus

1. Introduction

In recent times, cyber security is a key word in the automotive sector. It is necessary on the one hand to preserve the confidentiality, integrity and availability of such systems, and on the other hand, to ensure the safety of people. An example of such a need is visible in smart cities where the development of smart mobility [1,2] and autonomous driving involve different projects [3,4].

Driving and parking assistants, internet connectivity, and in general all the cyber-physical features installed in cars, involve risks in terms of security and privacy. One of many is the possibility to obtain personal information and vehicle data such as fuel consumption, oil level, and handbrake operation.

In the automotive sector, given the critical nature of the software installed on board, in addition to the approaches to continuous software improvement [5–7] and software security [8,9], it is necessary to have ad hoc methods and tools capable of managing the complexity [10] and criticality of cyber physical systems that include physical objects equipped with a microprocessor or microcontroller and

a memory bank. In this context, the so-called embedded system is the electronic control unit (ECU) [11] that governs various functions inside the vehicle.

The structure of automotive machines is divided into subsystems, each one in charge and separated from the others for each function, some of which have a clear impact on the safety of people on board: power train system; chassis control system; body control system; and infotainment system. By accessing the car's internal communication network, the controller area network (CAN) [12], it is possible to read the data exchanged between the ECUs, modify, replicate and/or block them, thus causing problems of various nature; from minor ones, such as providing the driver with incorrect information regarding some vehicle consumption data, to major ones such as damaging the car brakes by blocking messages before they reach the ECUs [13]. Since the CAN bus protocol does not provide security systems against cyber and physical attacks, and given the increasing number of novel types of attack, it is desirable to have a system to detect any kind of attacks. Traditional intrusion detection systems (IDS), normally used in network security, cannot be directly used for vehicular networks. Due to the high interest in the automotive sector in recent years, an effective and efficient IDS that can work for in-vehicle networks is an important need [14]. The purpose of the paper is to test an anomaly-based IDS implemented using an artificial neural network that, compared to other machine learning approaches, can show remarkable performance in detecting attacks on in-vehicle networks.

Given these premises, we propose a distance-based IDS for the identification of attack messages on an in-vehicle CAN bus. The system has been implemented using the Kohonen self-organizing map (SOM) network, an efficient and fast intrusion detection method that can be trained both as supervised and unsupervised learning. Due to its self-organizational and topological properties, the algorithm has powerful capabilities in the clustering and visualization of complex, highly dimensional data, and has found broad applications in security issues. In particular, the Kohonen SOM network has revealed remarkable performance as an anomaly detector in real-time systems. Compared to other intrusion detection methods, the Kohonen SOM network has shown better performance with a shorter training time and higher detection efficiency [15–17]. A recent review of the literature shows that the use of the SOM algorithm opened up new areas of security, which have not been tackled in a similar way before, such as anomaly-based detection [18]. In view of these considerations, we propose to test the Kohonen SOM network on an in-vehicle communication network, since, to the best of our knowledge, it was never tested in such domain.

In our research, we implemented an anomaly-based IDS for the identification of attack messages injected on the CAN bus using both supervised and unsupervised SOM networks [19] combined with a K-means clustering algorithm.

In this paper, we present only the results for the supervised SOM network comparing two different approaches. The first was a supervised X–Y fused Kohonen network (XYF), the second was a supervised XYF Kohonen network combined with a K-means clustering algorithm (XYF–K). The two methods were tested on an open source dataset concerning traffic data messages on a CAN network 2.0B and containing more than 2000 different attack types sent totally at random. The dataset contains a large volume of traffic data messages with a low number of features and highly imbalanced data distribution. Despite the complex structure of the CAN bus dataset, the proposed methods showed a high performance in the accuracy of detection of attack messages.

This paper presents the following contributions:

- We propose an anomaly-based IDS to detect attack injections on a CAN bus. To the best of our knowledge, this is the first work which tests the Kohonen SOM neural network as IDS for detection on in-vehicle networks. The SOM network can be implemented both as supervised and as unsupervised learning.
- We tested the proposed IDS on real data concerning in-vehicle network attacks. Data consists in four datasets, each containing a different attack type, in particular, *Denial of Service Attack* (*DoS)*, *spoofing the drive gear*, *spoofing the Revolution per Minute (RPM) gauge*, and *fuzzy* attacks. Data were obtained by injecting and logging CAN messages on a real vehicle [20]. Moreover, we present the effectiveness of the proposed method showing the performance metrics.

- Most of the researches proposed in the literature present detection methods able to identify only periodic malicious attacks, but not aperiodic violations [18]. The proposed IDS show a high performance in detecting attacks sent on the CAN bus with any kind of period, also aperiodic attacks.
- We compared the results obtained in terms of the system's accuracy to the other studies that tested different IDS approaches on the same dataset.
- This is the first work which tests an IDS not only on single datasets containing different attack types but also merging datasets into a unique dataset containing all attack types.

This paper is organized as follows: Section 2 illustrates the related works; Section 3 describes the CAN message structure; Section 4 explains the theoretical framework used in the work; Section 5 shows the experimental process; in Section 6 are pointed out the results and discussions; Section 7 sets out the conclusions.

2. Related Works

The attackers can maliciously interfere with vehicular communication that occurs both internally and externally to a vehicle [21]. Internal vehicular communications can take place within the in-vehicle network that is based upon the intercommunication of the many ECUs; instead, the external vehicular communications occur when the vehicle connect directly to USBs and maintenance tools. In this research work, we take into consideration the in-vehicle network with the objective of defining a model for the identification of attack messages on an in-vehicle CAN bus.

CAN transmissions are broadcast to all nodes and according to [22], the attacks could replace an authorized ECU program with an illegitimate, malicious program, and connecting to the CAN bus using an unauthorized device. Therefore, the attackers can perform a *Masquerading Attack* i.e., masquerades as a legitimate node because the CAN frame are not encrypted and it does not support message authentication [21]; *Eavesdropping Attack*, the attackers can intercept the CAN transmissions and identify patterns in valid CAN frames [23]; *Injection Attack* to inject false messages into an automotive bus system; *Replay Attack*, where valid frames are sent continuously in order to impede the vehicle's real-time functioning; *Denial of Service Attack* (DoS), which occurs when an attacker continually sends high priority messages that block legitimate low priority messages [21]; *Distributed Denial of Service (DDoS) Attack*, the attackers launch an attack from different locations and is more severe in the vehicular environment because the mechanism of the attack is in a distributed manner, where the impact is dispersed in the network [24]; *Bus-off Attack*, a type of DoS attack that exploits CAN error handling (bit error, stuff error, CRC error, form error, ACK error) to intentionally shift the victim node to the bus off state. The attacker causes a bit error of the victim node and a message collision by transmitting the frame at the same timing as the victim node. This is repeated every time the victim node sends a message, and puts the victim node into the bus-off state [25]; *Fuzzy Attack*, that is aimed to make the system confused thanks to the manipulation of normal CAN ID and data fields [26]; *Impersonate attack* is aimed to make legitimate engine speed signal (RPM) and gear components behave abnormally [26,27].

In order to identify attacks on in-vehicle communications, different intrusion detection systems have been developed. The categories of IDS used in the automotive domain comprise anomaly, specification, signature and hybrid-based techniques. Anomaly and specification-based methods detect anything that deviates from a normal behavior, but, in the first approach, normal behavior is learned in a training phase; in the second approach, specifications and rules need to be defined manually by a human expert. The signature-based approach detects attacks by using a set of previously defined signatures, rules and malicious events. Meanwhile, the hybrid-based approach combines several IDS techniques. Due to the characteristics of the CAN protocol, which can frequently be modified, an anomaly-based approach using the machine learning technique can be a suitable detection method. Indeed, it allows the IDS to learn from examples and, thus, to adapt to any change in the protocol with no need of the rules previously defined.

As a result, different machine learning techniques have been used, for example, neural network classifiers, Bayesian classifiers, Bayesian regularized neural networks, naive Bayes, support vector machines (SVM), and self-organization maps (SOM) [28,29].

In [30], an intrusion detection system (IDS) based on a deep convolutional neural network (DCNN) is proposed to protect the CAN bus of the vehicle. The DCNN learns the network traffic patterns and detects malicious traffic without hand-designed features. The proposed model demonstrated a good detection performance against message injection attacks. However, it has a fundamental limitation in detecting the unlearned types of attacks because it is based on supervised learning. Instead, in [31], the authors used an unsupervised deep neural network (DNN) to enhance the security. They trained the parameters of the DNN with probability-based feature vectors extracted from the in-vehicular network packets by using an unsupervised pre-training method of deep belief networks, followed by the conventional stochastic gradient descent method. The DNN provides the probability of each class to discriminate normal and hacking packets, and thus, the system can identify any malicious attack to the vehicle as a result.

DNN is also implemented in [32] treating anomaly detection as a cross-domain modelling problem. The proposed technology provides real-time responses to anomalies and attacks to the CAN bus, and improves the detection ratio.

Instead, in [33] a novel neural network architecture, CANet, is presented. It is trained in an unsupervised manner to detect intrusions and anomalies on the CAN bus. The risks highlighted by the authors are that neural networks require a large amount of training data and are typically more computationally expensive than many other approaches, and in a security setting, they can be sensitive to adversarial attacks. Additionally, in [26], an anomaly detection method using an unsupervised deep learning-based approach, deep contractive autoencoders (DCAEs), was developed. They imposed a different penalty term to the CAN data representation in order to encourage robustness towards small changes.

After analyzing the attacks and the different in-vehicle CAN bus solutions developed to identify them, our research focused on the implementation of an IDS based on a supervised Kohonen SOM network for the identification of attack messages, considering the following types: DoS, spoofing gear, spoofing RPM, and fuzzy.

The Kohonen SOM is a network that can be trained both as a supervised and an unsupervised learning network. There are different works that show the validity of this approach in IDSs. In [34], a SOM network was performed for real-time textual classification in order to filter web contents. A SOM network was also used for real-time intrusion detection in home appliances interconnected by a gateway [35]. Moreover, a SOM network was performed for real-time credit card fraud detection in order to analyze customer behavior [36]. The network showed high capabilities in identifying hidden patterns otherwise difficult to understand. In [37], a self-organizing feature map neural network combined with a K-means clustering algorithm (KSOM) was proposed to improve the efficiency of the intrusion detection network by leveraging the characteristic of the K-means clustering algorithm, such as a faster clustering speed. The model was tested on the NSL-KDD dataset, a classic well known IDS dataset, containing four attack categories and approximately 40 different attack types, and a high number of features, equal to 41, for each connection record. Experimental results showed that the iterations of the KSOM algorithm were significantly reduced compared with the SOM network model and the accuracy was significantly improved with respect to other methods of intrusion detection network.

Taking into account the strong capabilities of the SOM network as an anomaly detector in real-time systems, Kohonen SOM networks have been applied to many different areas of security [18] but have never been performed on an in-vehicle communication network.

The contribution of our research is to introduce a distance-based IDS using a Kohonen self-organizing map network integrated with a K-means clustering algorithm on a CAN bus for identifying attack messages on the bus.

Results and discussions for the supervised SOM are presented in this paper. We compared two different approaches. In particular, we first performed a supervised X–Y fused Kohonen network

(XYF) and then combined the XYF Kohonen network with a K-means clustering algorithm (XYF–K) in order to improve the efficiency of the intrusion detection network. The two procedures were tested on an open source car hacking dataset containing normal and attack messages in order to detect the attack messages injected on the bus. The dataset was characterized by a low number of features, just five for each message, and four attack categories. It contained messages sent on the CAN bus with different periods that were the time elapsed between sending two messages of the same type. The period could be regular, with a minimum interval time or a totally arbitrary time. The dataset presented a quite complex structure to analyse since it contained a high number of different attack types, more than 2000, with just one occurrence, sent totally at random. Olufowobi et al. [38] implemented an IDS based on a supervised learning approach and tested the model using the described dataset but only including the messages sent at regular or with a minimum interval time. We included in the analysis all kinds of messages and despite the complex structure of the CAN bus dataset, the proposed methods showed a high performance in identifying even the attack messages sent totally at random at arbitrary times on the CAN bus.

3. Control Area Network (CAN)

The CAN bus is a message-based protocol designed for in-vehicle networks. It enables numerous electronic components, throughout the in-vehicle system, to communicate with each other through a single/dual-wire bus [29]. Consequently, all the nodes in a CAN bus can receive all packet transmissions and a frame is defined as a structure [30]. Figure 1 shows the standard structure, a sequence of CAN data (bytes) in the network.

Figure 1. Controller area network (CAN) message structure.

The meaning of each field in a CAN frame is: SOF (start of frame), which specifies the start of a new message with a dominant bit to synchronize all the nodes for CAN message transmission; Arbitration includes 11 bits and can be extended up to 29 bits. The ID field for each transmitted CAN frame indicates the packets' priority. The lower the ID bits, the higher the priority of the packet the value signifies, and the single remote transmission request (RTR) bit is dominant when information is required from another node; Control provides information in order to verify that all the expected packets are received successfully by the receiver. It consists of three fields: the Identifier Extension (IDE) must be recessive for extended format and dominant for standard format, Reserved Bit (RB0) must be dominant for standard format, and DLC (data length code) contains the number of data bytes; Data contains information for CAN nodes to perform actions; CRC (cyclic redundancy check) contains the checksum of the previous data for error detection and in CRC Delimiter (CRC Del) the single bit must be recessive; ACK (acknowledge) ensures that all the nodes involved in the message receive everything correctly, and in case of error, that the transmitter is immediately alerted to send the data packets again. In ACK Delimiter (ACK Del) the single bit must be recessive; EOF (end of frame) indicates the end of a current CAN Message.

4. Theoretical Framework

4.1. Unsupervised Kohonen Network

The Kohonen self-organizing map is a type of artificial neural network (ANN). This technique allows the visualization of high dimensional data reducing the dimensions of data to a map. The Kohonen SOM is a nonlinear mapping network aimed at detecting the similarities among the data in the input layer and at representing them in an output layer of interconnected neurons. Input data are assigned to neurons according to the defined measures of distance between them [39].

Unlike other techniques in neural networks, SOMs can classify data without supervision, that is to say the model does not require any target variable. It is an unsupervised learning model and differs from other artificial neural networks because it cannot adjust the weights using back propagation methods for error-correction learning. Instead, the SOM network has a feed-forward structure with a single computational layer [40] and uses competitive learning in order to find similarities among the data, clustering them into different classes of data [41]. The Kohonen neural network maps input data on a map of neurons according to spatial constraints.

The Kohonen output network consists of a competitive layer where a *n*-dimensional *codebook* vector is assigned to each neuron in the map and the elements of the vector represent the weights [42]. The spatial constraints also imply that neighboring neurons have similar codebook vectors. If two high dimensional input data vectors are similar, then they will be mapped in nearby neurons on the network grid. Hence, input data vectors mapped on the same neuron or in the neighboring ones are similar. SOM networks try to reproduce the topological order of input data through clusters of neurons and neighbors whose number is defined by the size of the map [43].

The SOM algorithm computes similarities between each input data vector and the neurons codebook vectors in order to find the most similar. This neuron is the winner, called the *best matching unit* (BMU), and it updates its weights vector using a weighted average in order to better match with the input vector. The weight of the attraction of the BMU to the input data vector changes at each iteration and it is one of the training parameters of the model also called *learning rate α*. The learning rate α decreases during the training process ensuring the convergence of the model [43]. Moreover, the neighboring neurons update their weight vectors moving closer to the input and ensuring the spatial constraints in order to preserve the topology of the map.

A simple description of the algorithm can be explained as follows. After defining the number of neurons in the Kohonen SOM network, a codebook vector is initialized for each neuron. Then, the Euclidean distance is computed between a CAN message vector randomly selected and all the neurons in the Kohonen layer. The CAN message vector is assigned to the neuron with the minimum value of distance, called the winning neuron or best matching unit (BMU). The neurons with the most similar codebook vectors are identified as neighboring neurons. The BMU and the neighboring neurons are updated in order to better match to the CAN message vector. This procedure is repeated for all the CAN message vectors and for a given number of iterations.

Two main algorithms are used to train the SOM network: the online and the batch algorithms [44]. The difference is that in the *online* algorithm, the BMU and the neighboring neurons are updated immediately after an input data vector is presented to the network, whereas in the *batch* algorithm, codebook vectors are updated after all the input data vectors are presented to the network [45].

Formally, we defined the input data vectors X_i, with $i = 1,…,n$, the number of Kohonen neurons R_j, with $j = 1,…,m$, and set the number of iterations t with $t = 1,…,z$ [46]. The number of iterations in the training process is expressed in *epochs*, where one epoch indicates the steps of the training algorithm which allows one complete presentation of all the input data vectors to be learned by the network. The online training algorithm of the SOM can be implemented using a stepwise recursive procedure detailed in Algorithm 1.

Algorithm 1 Kohonen SOM algorithm

Input: Input layer consisting of CAN message data vectors X_i with $i = 1,…,n$,

Output: Output Kohonen layer R_X containing final codebook vectors associated to neurons R_j with $j = 1,…,m$,

Results: Assign CAN message data vectors X_i to the winning neuron R_{BMU} on the Kohonen map

set(n, m, z)//set number of messages, number of neurons, number of epochs
$R_j \longleftarrow \emptyset$
for $j = 1,…,m$ **do**
　　$R_j \longleftarrow$ random(X_i)//random codebook vectors initialization
end for
set(α)　//set initial learning rate

while $t < z$ **do**
　　for $i = 1: n$ **do**
　　　　$X_i \leftarrow$ random(X_i)//random selection of a CAN message data vector
　　　　for $j = 1: m$ **do**
　　　　　　$d_j = \|X_i - R_j\|$ //compute Euclidean distance
　　　　end for
　　　　$d_c = \min_j d_j$ //compute the winning neuron R_{BMU}
　　　　for $j = 1: m$ **do**
　　　　　　$R_j = R_j + h_{jc}(\alpha)[X_i - R_j]$ //update neurons
　　　　end for
　　end for
　　update α
end

Many different measures of similarity can be used to measure the distance between the input data vectors X_i and the neurons R_j codebook vectors [43] such as Manhattan, Tanimoto, Bray Curtis, Canberra, Chebyshev distances. However, Euclidean distance normally gives slightly better classification results [47]. We define $R_c(t)$ as the winning neurons R_{BMU} with the minimum Euclidean distance from the input data vector $X_i(t)$ at the iteration t where:

$$c = \min_j \{\|X_i(t) - R_j(t)\|\} \qquad (1)$$

$R_c(t)$ is the neuron which best matches the input vector $X_i(t)$ in the grid at iteration t [48]. Once the winning neuron is found, the neuron and its spatial neighbors are updated by the following:

$$R_j(t + 1) = R_j(t) + h_{jc}(t)[X_i(t) - R_j(t)] \qquad (2)$$

where $h_{jc}(t)$ is the *neighborhood function*. The rate of change at different neurons around the winning neuron depends on the mathematical form of the neighborhood function. This function has a very central role in the SOM networks since it preserves the topological properties of the inputs. A variety of neighborhood functions can be used, but the most applied in SOM neural networks is the *Gaussian neighborhood function*:

$$h_{jc}(t) = \alpha(t) \cdot \exp\left(\frac{\|r_j - r_c\|^2}{2\sigma(t)^2}\right) \qquad (3)$$

where $\alpha(t)$ is the *learning rate function* which is a function monotonically decreasing at each iteration t and $\sigma(t)$ expresses the width of the neighborhood function also decreasing with the training process. r_j is the position of neuron j.

A variant of the traditional online SOM algorithm is the *batch* algorithm where neurons' codebook vectors are adjusted only after all the input data vectors X_i in the input layer are assigned to their winning neurons R_{BMU} in the network. At iteration t, the codebook vectors of winning neuron and neighbors' neurons are updated as follows:

$$R_j(t + 1) = \frac{\sum_{i=1}^n h_{jc(i)}(t)X_i}{\sum_{i=1}^n h_{jc(i)}(t)} \qquad (4)$$

where $c_{(i)}$ is the index of the winning neuron R_{BMU} for the input data vector X_i and n is the number of the input data vectors.

4.2. Supervised Kohonen Network

The Kohonen neural network is broadly known as an unsupervised learning model, although it can be used as a supervised classifier. Further information, such as class information, can be entered in the network with the aim of modeling the relationship between the input data and a dependent variable for predictive purposes [45].

In the supervised SOM networks, two layers are presented to the network, one for the input data and one for the dependent variable. We define X_i and Y_i, with $i = 1,…,n$, respectively, the input data and the dependent variable vectors, and $R_{X,j}$ and $R_{Y,j}$, with $j = 1,…,m$, the Kohonen neurons, respectively, in the R_X and in the R_Y map. The dependent variable Y layer has the same topology and the same numbers of neurons in the grid as the input data layer X, but a different number of weights in the codebook vectors. The neurons codebook vectors $R_{X,j}$ and $R_{Y,j}$ have the same number of weights, respectively, of the input X_i and the output Y_i data vectors.

The supervised Kohonen network can be implemented following different approaches [49]. The training process is the same as the unsupervised network.

The first approach is based on the counter propagation network (CPN), similar to the unsupervised network. The input data X_i and the dependent variable Y_i vectors are trained simultaneously but separately. The dependent variable layer Y does not influence the training process.

In the X–Y fused (XYF) Kohonen approach, the network is trained on a concatenated layer of input data X_i and dependent variable Y_i vectors. A fused similarity measure is computed as a weighted combination of separate distances between the input data vectors X_i to the neuron codebook vectors $R_{X,j}$ and the corresponding dependent variable Y_i vectors to the neuron codebook vectors $R_{Y,j}$. $R_{X,j}$ and $R_{Y,j}$ are updated simultaneously. At the iteration t, the fused similarity measure $D(i,j)$ for the observation (X_i, Y_i) can be expressed as follows:

$$D(i,j) = \emptyset(t)D\big(X_i, R_{X,j}\big) - (1 - \emptyset(t))D\big(X_i, R_{Y,j}\big) \tag{5}$$

where $\varphi(t)$ is a weight which linearly decreases with iterations. The winning neuron is the one with the minimum $D(i,j)$.

The bidirectional Kohonen (BDK) approach uses the same training procedure of the XYF network with the difference that, first, only the codebook vectors $R_{X,j}$ in the R_X grid are updated. Then, the codebook vectors $R_{Y,j}$ in the R_Y grid are adjusted in a bidirectional way instead of a simultaneous way.

At the end of the training process, the output of the supervised Kohonen network consists of two separate neuron maps: R_X for the input data X and R_Y for the one dependent variable Y. The prediction for a new object is determined by first computing the winning neuron $R_{X,BMU}$ in the R_X grid and then assigning to the object the class of the corresponding neuron $R_{Y,BMU}$ in the R_Y grid.

4.3. The Proposed Methodology

In this paper, we propose an anomaly-based IDS for the identification of attack messages on an in-vehicle CAN bus. Firstly, we implemented a supervised XYF Kohonen approach to classify attack and normal messages sent on the CAN bus, obtaining good performance in detecting attack messages. The XYF Kohonen network is a power tool for intrusion detection, but it achieves its best performance when combined with other approaches, in particular, clustering algorithms [18]. The K-means algorithm is a clustering algorithm [50], which helps to achieve local clustering and shows high performance in dealing with big datasets. Previous works show how the SOM network coupled with the K-means clustering algorithm achieves better local optimal clustering results, improving the detection rate [51]. Thus, secondly, we implemented a hybrid approach combining the supervised XYF Kohonen network with the K-means clustering algorithm in order to improve the performance of the model in correctly identifying the attack messages injected on the CAN bus. In this approach, the clustering problem is not solved by the XYF network that only calculates the codebook vectors assigned to each neuron. The codebook vectors are, then, used as an input of the K-means algorithm for local clustering in order to achieve clustering results. This approach enables solving a lack of the XYF network in correctly classifying some observations, improving the detection accuracy and

reducing the false negative rate. This measure has great importance in attack detection for in-vehicle CAN buses since undetected attacks can undermine vehicle safety.

5. Experimentation

The research goal addressed in this paper is to implement an anomaly-based IDS for the identification of attack messages on an in-vehicle CAN bus. To this aim, we compared two methods on four different CAN bus attack datasets. The first method was a supervised XYF Kohonen network, and the second was a supervised XYF Kohonen network combined with a K-means clustering algorithm in order to improve the performance of the model and correctly identify the attack messages injected on the CAN bus.

The research goal and research questions were defined according to the goal–question–metric (GQM) paradigm [52,53] as follows:

- **Research goal:** Analyze car hacking datasets in order to test the efficiency of a supervised Kohonen SOM network as an anomaly detector for a in-vehicle CAN bus.

- **Research questions:**

RQ1: Can the supervised Kohonen SOM network be used as an anomaly detector for in-vehicle communication networks?

RQ2: How does the proposed method perform in terms of the system's accuracy compared to other IDS approaches to detect attack messages on the CAN bus?

- **Metrics:**

M1: Given a car hacking dataset, the efficiency of the XYF and the XYF–K algorithms was tested computing the following classification metrics: accuracy, detection rate, false positive rate (FPR), precision, and F1-score.

M2: Given a car hacking dataset, the accuracy of the XYF–K algorithm is higher or, at least, equal compared to the accuracy of other IDS approaches, also including the analysis of periodic and aperiodic CAN messages.

For this aim, the experimentation was performed in three main steps:

1. Data preprocessing: preprocessing of the car hacking datasets concerning the traffic data messages on a CAN bus and containing the injection of attack messages. The datasets regard different kinds of attack including *DoS* attack, *spoofing the drive gear*, *spoofing the RPM gauge* and *fuzzy* attack. The datasets are available at [20].
2. Experimental process: implementation of the XYF Kohonen network and of the XYF–K algorithm first on each dataset separately and then merging all of them in a single *mixed* dataset including all kinds of attack. The network was performed on random samples of 10,000 split into the training set and test set, for each dataset and for the *mixed* one. The training set was used to train the network and the test set for prediction.
3. Evaluation metrics [54]: evaluation of the performances of the two models.

5.1. Data Description

We analyzed the open source car hacking data which included four datasets: *DoS* attack, *spoofing the drive gear*, *spoofing the RPM gauge*, and *fuzzy* attack. The datasets were created by logging the CAN network 2.0B traffic via the OBD-II port from a real vehicle, while message injection attacks were performing [27,30]. Each dataset contained 300 intrusions of message injection for 3 to 5 s time lapse with a total of 30 to 40 min of CAN traffic.

In particular, in the *DoS* attack database attack messages of 29 zero bits CAN ID, the most dominant in the CAN bus were injected every 0.3 milliseconds; the aim was the tampering of the CAN network accessibility. Since the most dominant CAN ID wins the bus, all other ECUs are precluded to access the network and send messages.

In the *spoofing gear/RPM* databases, the attack messages of CAN IDs related to gear/RPM information were injected every 1 millisecond. The injected messages transmitted information about

the driver gear and the RPM gauge, respectively, changing the original status on the instrument panel. Data were labeled for normal and attack traffic data.

In the *fuzzy* attack database, the attack messages were similar to the *DoS* attacks, but CAN IDs and DATA values were injected totally at random every 0.5 milliseconds impairing the vehicle functionality.

An overview of the dataset is shown in Tables 1 and 2.

Table 1. Car hacking data type and size.

Attack Type	N. of Messages	% of Normal Messages	% of Attack Messages
Denial of Service (DoS) Attack	3,665,771	0.84	0.16
Fuzzy Attack	3,838,860	0.87	0.13
Spoofing the Drive Gear	4,443,142	0.87	0.13
Spoofing the Revolutions per Minute (RPM) Gauge	4,621,702	0.86	0.14

All the datasets have the same structure and include: *timestamp*, recorded time in seconds, *CAN ID*, identifier of CAN message in HEX, *DLC*, number of data bytes from 0 to 8, *DATA [0~7]*, data values (byte), and *labels* T or R where T represents the injected attack messages while R represents the normal messages.

Table 2. Car hacking dataset overview.

TimeStamp	CanID	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Label
1478193191230940	04f0	8	0	0	0	80	0	67	d1	13	R
1478193191234870	05f0	2	1	0							R
….	…	…	…	…	…	…	…	…	…	…	…
1478193191271620	043f	8	1	45	60	ff	6b	0	0	0	T
1478193191272270	316	8	5	23	70	9	23	21	0	70	R
….	…	…	…	…	…	…	…	…	…	…	…

5.2. Data Preprocessing

Before running the model, we first preprocessed the data. A CAN message is characterized by its period and by its ID, which represents a unique identifier and expresses the message's priority. Data processed in the network were the CAN ID identifier, data values DATA [0~7], DLC, period, and label.

We define some notations: at each timestamp S_i with $i = 1,…,n$, where n is the total number of messages, an ECU sends a message to the CAN bus. Let $I_k = \{I_1,…,I_r\}$ be the set of all CAN ID identifiers in the dataset. For each I_k, with $k = 1,…,r$, we have a set of $D_{Ik} = \{D_{Ik_1},…, D_{Ik_s}\}$ data values, and s can take different values in each set D_{Ik}.

CAN ID identifiers and data values were regarded with a semantic approach considering each CAN ID identifier I_k as a class of messages sent by an ECU and the data values D_{Ik} as the related value information [32]. Since the Kohonen network works by using numerical data, we used the one-hot encoding technique to turn categorical data into a matrix representation. In particular, each CAN ID unique identifier was represented by one column with value 1 if the CAN message instance belongs to that CAN ID, or otherwise 0 (Table 3). The data values DATA [0~7] were merged in a single string representing the value information related to a particular CAN ID message. This new categorical variable was also turned into a matrix representation using the one-hot encoding technique.

We did not process the CAN message data in a sequential way but with respect to their period. CAN messages can be periodic, sporadic, or aperiodic. Periodic messages are sent at regular time intervals, sporadic messages occur with a minimum time interval and aperiodic messages at arbitrary times [19]. Hence, starting from timestamp defined as S_i, with $i = 1,…,n$, we calculated a new variable T whose elements T_i, with $i = 1,…,n$, express the time in milliseconds between two successive occurrences with the same CAN ID identifier. The dataset contained a high number of periodic, sporadic and even aperiodic messages with different CAN IDs sent at different times and with different frequencies. They all were included in the analysis.

Finally, we normalized all the numerical variables in order to avoid bias in the training process that can be generated when dealing with very large input vectors [42]. Normalization was obtained using a linear transformation to scale the numerical variables to have values between 0 and 1.

X represents the input layer processed in the Kohonen network where X_i is the *i*-th input CAN message data vector (Table 3).

A second layer Y, given by the dependent variable label, was presented to the network. The variable label, since it is a categorical variable with the classes "Attack" and "Normal", was also transformed into a matrix with two columns, one for each class, and *n* rows of 0 or 1, one for each input data vector (Table 3).

Table 3. Processed dataset.

Input X Data Vector	D LC	04f 0	05f 0	043 f	...	000800 67d113	14560ff 6b000	...	Time	Dependent Variable Y Vector	Normal	Attack
										Dependent Variable Y Layer		
X_1	1	1	0	0	...	0	0	...	0.27	Y_1	1	0
X_2	0.1	0	0	1	...	0	1	...	0.84	Y_2	1	0
X_3	1	0	1	0	...	1	0	...	0.97	Y_3	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
X_n	0.5	0	0	0	...	0	0		0.01	Y_n	1	0

The *DoS* and *spoofing the gear/RPM* datasets present a quite regular structure. Normal messages are sent on the CAN bus using 26 different CAN ID identifiers. In a *DoS* attack, messages are only sent using one different CAN ID, in *spoofing the gear/RPM*, one of the 26 CAN IDs that send normal messages also send attack messages. These three datasets have a quite simple structure and the proposed method tested on them completely succeeded in detecting the attack messages. The *fuzzy* dataset shows a more complex structure, since the messages are sent with 2017 different CAN IDs, 37 of which also send normal messages. Moreover, the messages sent using the 97.1% of unique CAN IDs show a relative frequency less than 0.1% and are transmitted totally at random. Hence, the *mixed* dataset, obtained merging the four datasets, presents a higher complexity for the purpose of the analysis. The number of CAN ID identifiers is shown in Table 4.

Table 4. Number of CAN ID identifiers (total, normal and attack) in each dataset.

Attack Type	N. of Total CAN ID	N. of Normal CAN ID	N. of Attack CAN ID	N. of Both Normal and Attack CAN ID
DoS Attack	27	26	1	-
Fuzzy Attack	2017	37	1980	2017
Spoofing the Drive Gear	26	26	1	1
Spoofing the RPM Gauge	26	26	1	1

5.3. Experimental Process

After preprocessing the data, we performed a supervised XYF Kohonen network with the aim of configuring a network able to correctly identify attack messages injected on the CAN bus. The goal was to assign attack messages and normal messages to different neurons on the map since the Kohonen network provides a distance-based output assigning input data vectors based on their global and local similarity [39].

We used a process of trial and error to define the parameters of the supervised XYF Kohonen network. In large maps, searching for winner neurons and updating the information are time-consuming operations. If the neurons in the network are initialized with weight vectors close to the final codebook vectors, the convergence of the model requires to be at least an order of magnitude faster [48]. Starting the training process with a small map allows to calibrate the network in a low number of epochs, even with an unstable convergence. The codebook vectors assigned to the neurons

in this first training of the network can be used to properly initialize a larger map that will converge in a longer training process but reducing time consumption. Accordingly, we started to train the XYF network on a 2 × 2 hexagonal map.

We performed the XYF network using the online SOM algorithm which returned higher levels of accuracy than the batch algorithm trained on a 2 × 2 neurons map for a low number of epochs equal to 100. The initial value of the learning rate α was set to 0.5, linearly decreasing to 0.001 at the end of the training. The weights φ parameter of the fused similarity measure $D(i,j)$ was set equal to 0.1 for the R_X map and equal to 0.9 for the R_Y map in order to put more emphasis on the labels variable Y and enforce the spatial clustering of the attack and normal messages. The Gaussian neighborhood function was used to select the neighboring neurons around the winning neuron. The similarity between the input message data vectors and the SOM grid neurons was computed using the Euclidean distance since it retuned the best results.

The network perfectly separated the attack and normal input message data vectors in two distinct neurons in a low number of epochs. Results are shown in Section 6.

After training the XYF network, we expanded the grid introducing in the 2 × 2 neurons hexagonal map new interstitial nodes to form a larger map. In particular, we doubled in two times the size of the original array obtaining, first, a 4 × 4 neurons and then, an 8 × 8 neurons hexagonal map. The XYF networks on the expanded maps were performed initializing the neurons codebook vectors in the R_X and R_Y maps using the neurons codebook vectors returned in output by the XYF network previously trained. The networks were trained this time using the batch algorithm, since it is faster and computationally more efficient for longer training process. In fact, neurons' codebook vectors are updated only after the complete dataset has been presented to the network [45]. In order to speed up the process, the networks were run using a parallel implementation of the process on multiple cores. The XYF networks trained on both the expanded maps greatly improved the performance and converged in a training process of 500 epochs. Results are shown in Section 6. The XYF networks trained on different map sizes were used to predict the new input message data vectors.

The XYF network showed a powerful performance in classifying the attack and normal input data messages. Nevertheless, we combined the output of the supervised XYF Kohonen network with a K-means clustering algorithm (XYF–K) in order to improve the performance of the model in correctly identifying the attack messages injected on the CAN bus. The codebook vectors of neurons $R_{X,j}$ in the R_X map at the end of the Kohonen training procedure were used as input vectors of the K-means algorithm in order to divide the neurons in two disjoint clusters, normal and attack (Figure 2). Finally, input data message vectors assigned to the neurons belonging to the normal cluster, will be classified as normal messages, input data message vectors assigned to the neurons belonging to the attack cluster will be classified as attack messages. The number of clusters was set equal to 2 in order to cluster neurons and the associated CAN messages in two groups, attack and normal. The initial centroids were generated randomly resampling the initial starting point 30 times, in order to make the k-means algorithm converge to a global optimum. The initial centroids with the lowest sum of squares were finally chosen. The algorithm converged setting a maximum number of iterations equal to 30.

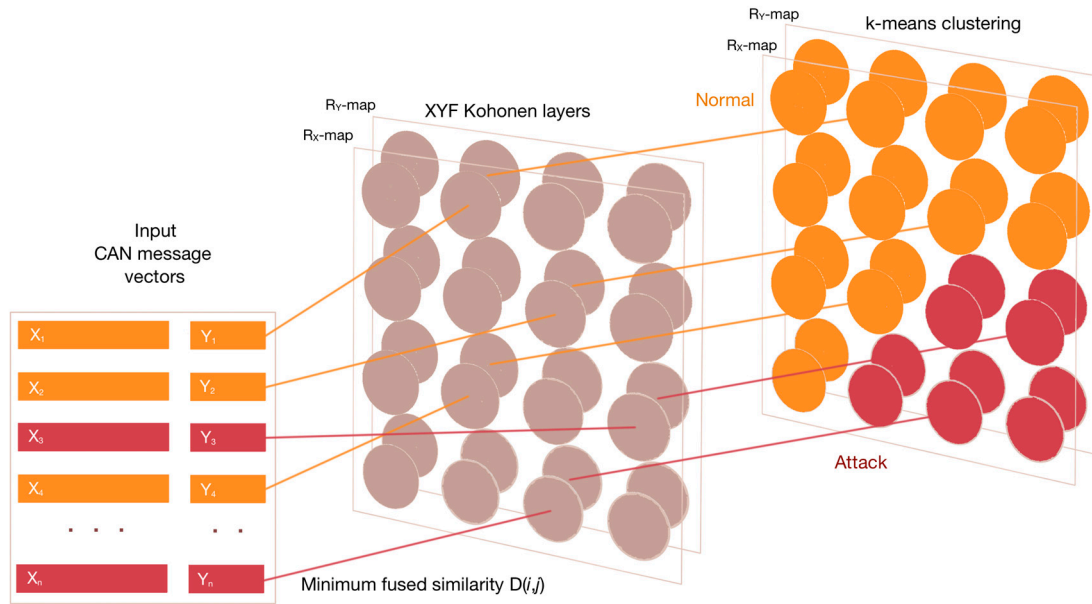The Kohonen neural network was implemented using the R project available at the repository: http://cran.r-project.org.

Figure 2. The XYF–K algorithm.

5.4. Evaluation Metrics

In order to evaluate the performances of the supervised Kohonen network implemented on different datasets, we computed the traditional classification metrics as accuracy, detection rate, false positive rate (FPR), precision, and F1-score [55].

We also computed the false negative rate (FNR) which represents the fraction of undetected attacks as follows:

$$FNR = \frac{FN}{TP+FN}, \tag{6}$$

where TP (true positive) and TN (true negative) are the number of messages correctly classified, respectively, as attack or normal, and FP (false positive) and FN (false negative) are the number of messages incorrectly classified, respectively, as attack or normal.

Since even a very small number of undetected attacks can determine tampering in the vehicle, undermining its safety, the FNR measure has great importance in attack detection for in-vehicle CAN buses and its value should be very small.

6. Experimental Results and Discussion

As explained in Section 5, in the experimentation we performed two methods, the supervised XYF Kohonen network and the supervised XYF–K algorithm, on the *DoS*, *spoofing gear/RPM*, and *fuzzy* datasets in order to identify attack messages on the CAN bus. We processed the samples of 10,000 records, first, for each single dataset separately, then, for the *mixed* dataset including all kinds of attacks. The samples were randomly selected balancing the ratio of CAN ID messages and were randomly split into 80% training and 20% test set, again balancing the ratio of CAN ID messages both in training and in test set for all the datasets with the exception of the *fuzzy* dataset. The *fuzzy* dataset was split balancing the ratio of labels attack/normal because of the high number of different CAN ID messages (2017), the 97.1% of whom show a relative frequency less than 0.1% in the dataset and are transmitted totally at random. The training set was used to train the XYF Kohonen network and the XYF–K algorithm and the test set to evaluate the models' performance.

We started training the XYF Kohonen network on a 2 × 2 neurons hexagonal map. The network assigned a codebook vector to each neuron in the R_X and R_Y grids. The element of the neuron's codebook vector with the maximum value in the R_Y map determines the label of that neuron. The neurons in the R_X map assume the labels of the neurons located in the same position in the R_Y map.

Figures 3–7 show the results of the XYF Kohonen network, respectively, for the DoS, spoofing gear/RPM, fuzzy and mixed datasets. In addition, these results show that the supervised Kohonen SOM network can be used as an anomaly detector for an in-vehicle communication network (**RQ1**).

In the following discussion, we will refer only to the DoS dataset in Figure 3, since analogous considerations can be extended to the other datasets.
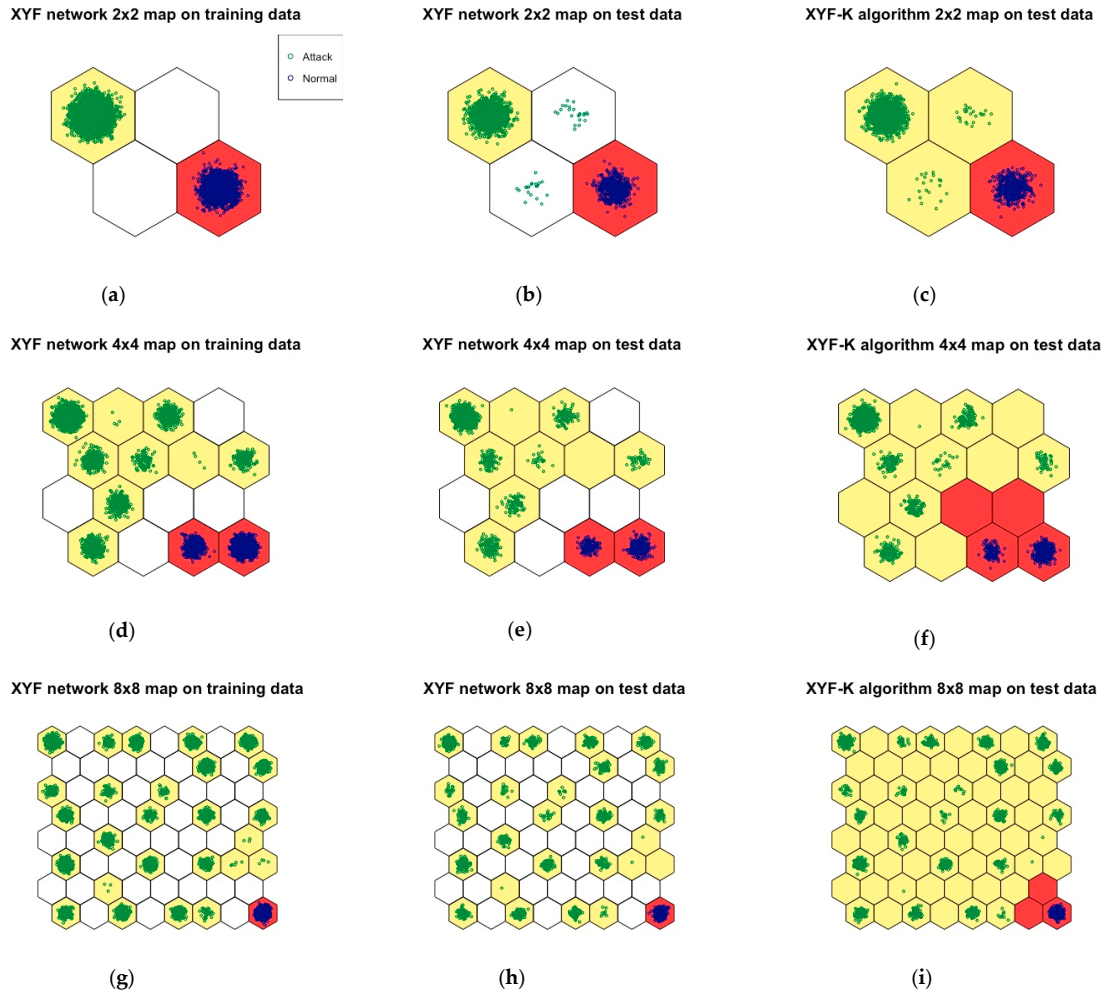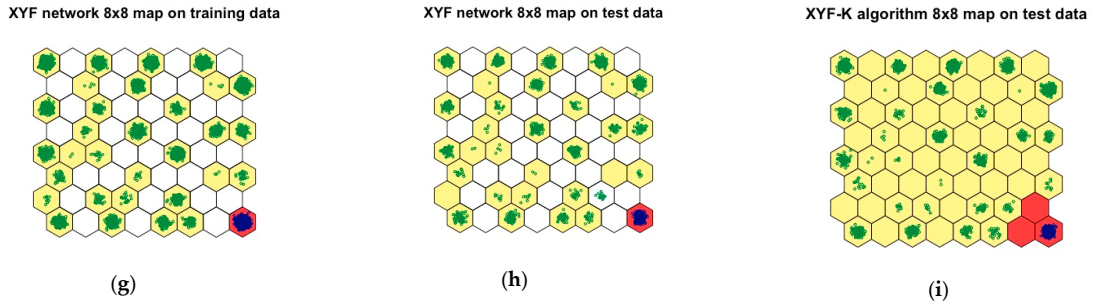

Figure 3. XYF network and XYF-K algorithm on the different neurons map sizes for the *DoS* dataset. (**a**) XYF network 2 × 2 map on training data; (**b**) XYF network 2 × 2 map on test data; (**c**) XYF-K algorithm 2 × 2 map on test data; (**d**) XYF network 4 × 4 map on training data; (**e**) XYF network 4 × 4 map on test data; (**f**) XYF-K algorithm 4 × 4 map on test data; (**g**) XYF network 8 × 8 map on training data; (**h**) XYF network 8 × 8 map on test data; (**i**) XYF-K algorithm 8 × 8 map on test data.
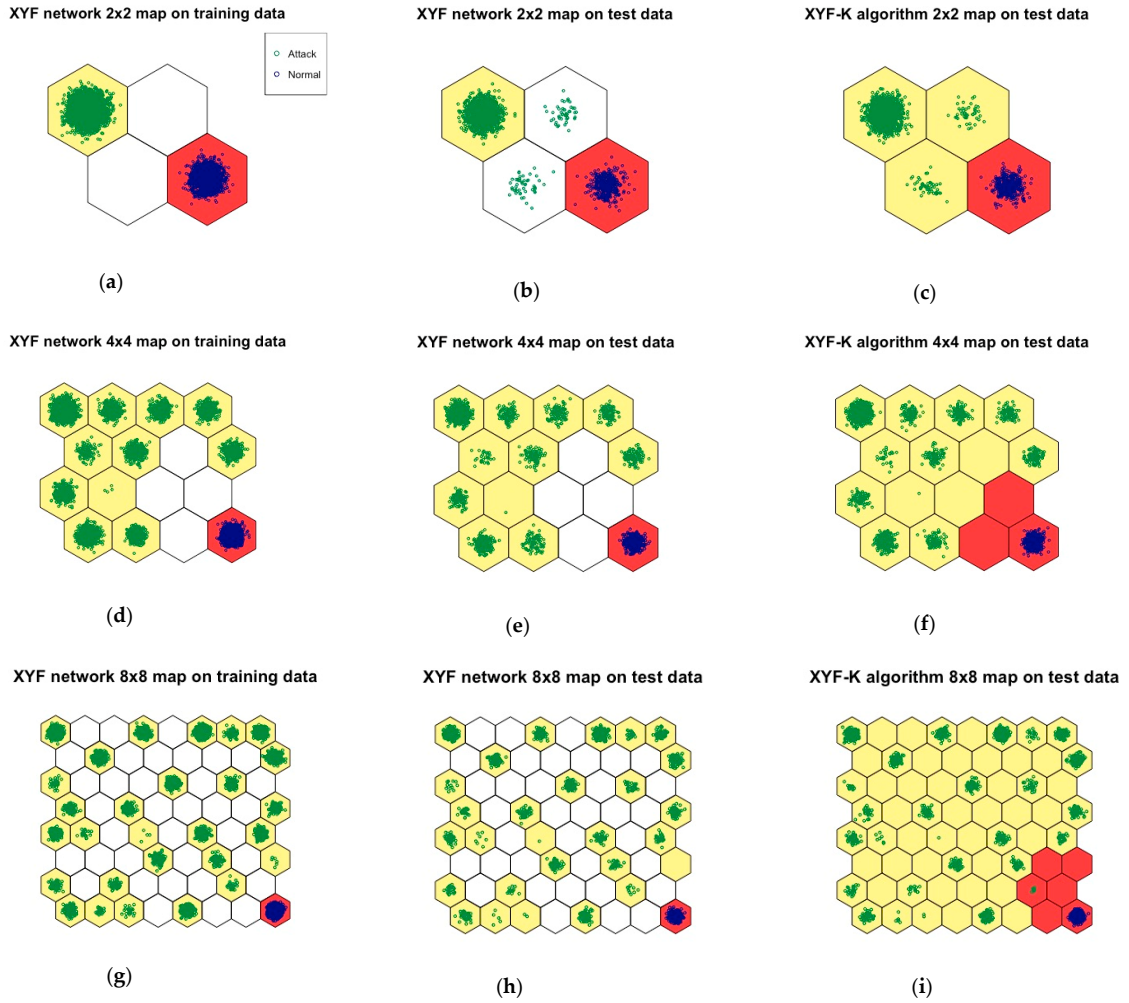
Figure 4. XYF network and XYF−K algorithm on the different neurons map sizes for the *spoofing gear* dataset. (**a**) XYF network 2 × 2 map on training data; (**b**) XYF network 2 × 2 map on test data; (**c**) XYF-K algorithm 2 × 2 map on test data; (**d**) XYF network 4 × 4 map on training data; (**e**) XYF network 4 × 4 map on test data; (**f**) XYF-K algorithm 4 × 4 map on test data; (**g**) XYF network 8 × 8 map on training data; (**h**) XYF network 8 × 8 map on test data; (**i**) XYF-K algorithm 8 × 8 map on test data.
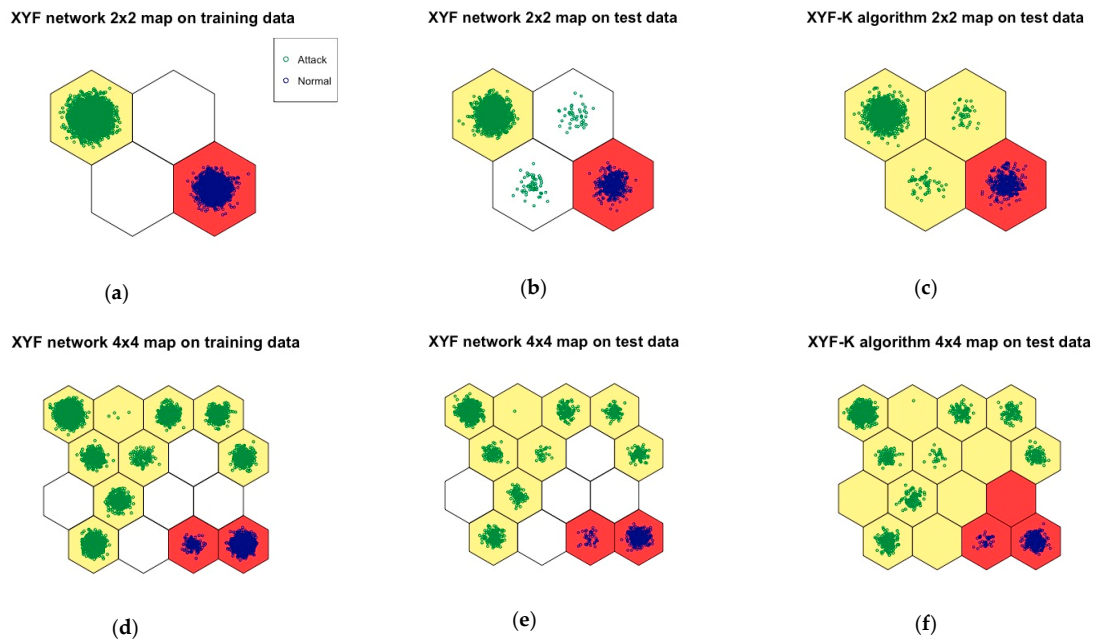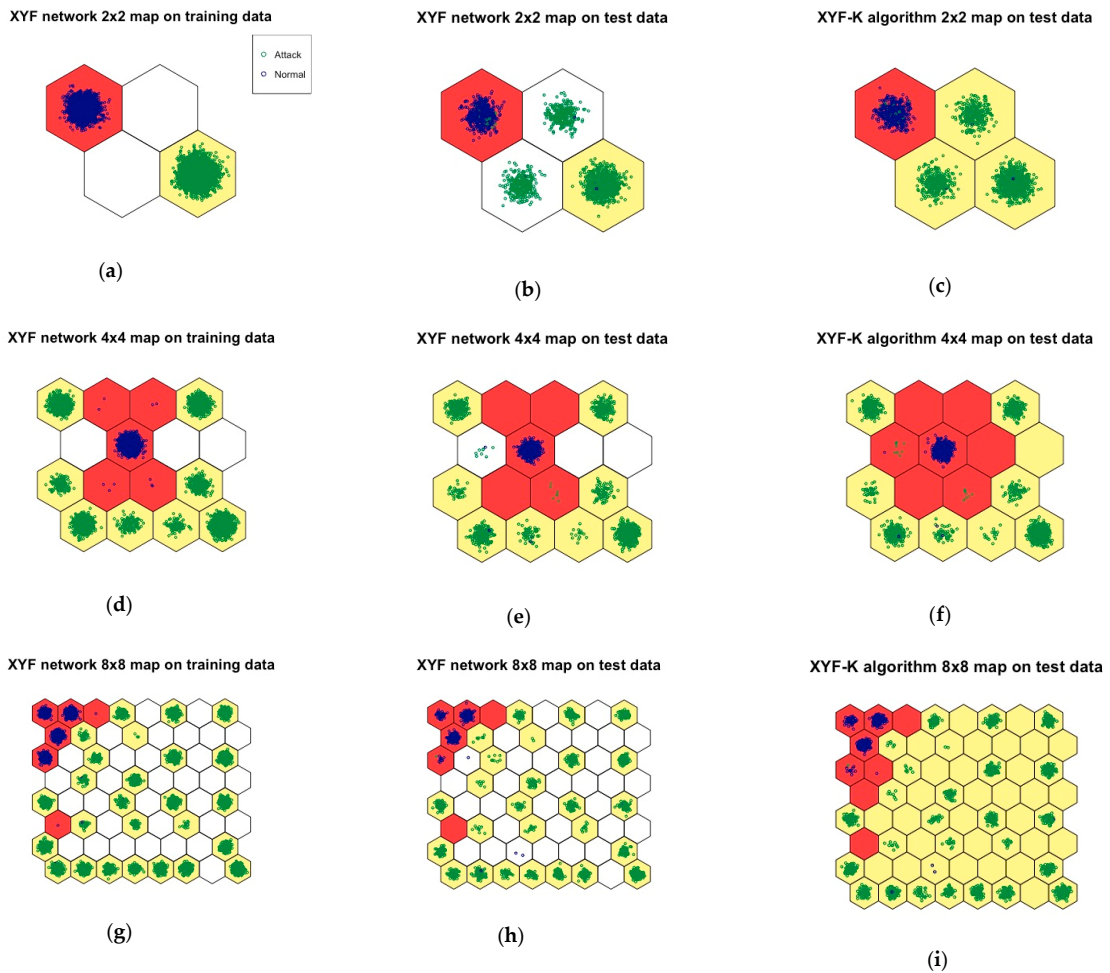
Figure 5. XYF network and XYF−K algorithm on the different neurons map sizes for the *spoofing RPM* dataset. (**a**) XYF network 2 × 2 map on training data; (**b**) XYF network 2 × 2 map on test data; (**c**) XYF-K algorithm 2 × 2 map on test data; (**d**) XYF network 4 × 4 map on training data; (**e**) XYF network 4 × 4 map on test data; (**f**) XYF-K algorithm 4 × 4 map on test data; (**g**) XYF network 8 × 8 map on training data; (**h**) XYF network 8 × 8 map on test data; (**i**) XYF-K algorithm 8 × 8 map on test data.

Figure 6. XYF network and XYF−K algorithm on the different neurons map sizes for the *fuzzy* dataset. (**a**) XYF network 2 × 2 map on training data; (**b**) XYF network 2 × 2 map on test data; (**c**) XYF-K algorithm 2 × 2 map on test data; (**d**) XYF network 4 × 4 map on training data; (**e**) XYF network 4 × 4 map on test data; (**f**) XYF-K algorithm 4 × 4 map on test data; (**g**) XYF network 8 × 8 map on training data; (**h**) XYF network 8 × 8 map on test data; (**i**) XYF-K algorithm 8 × 8 map on test data.
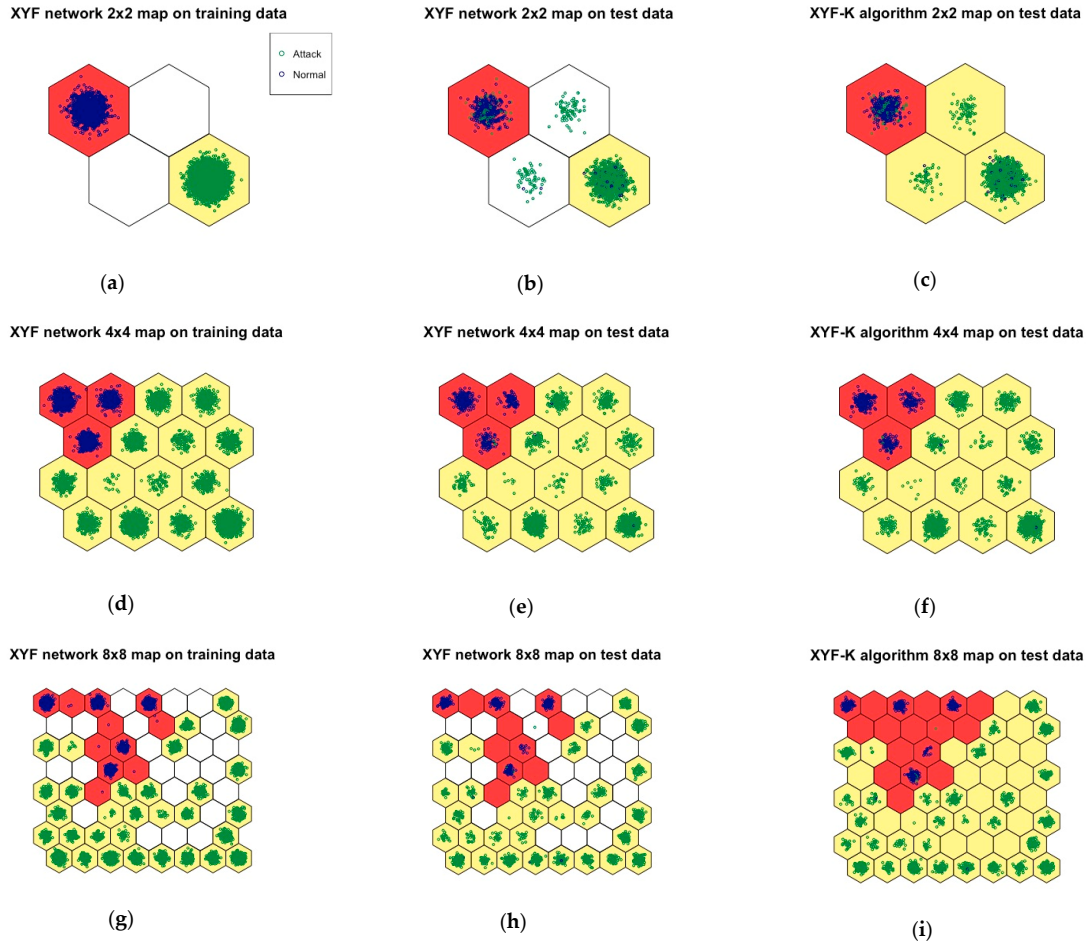
Figure 7. XYF network and XYF-K algorithm on the different neurons map sizes for the *mixed* dataset. (**a**) XYF network 2 × 2 map on training data; (**b**) XYF network 2 × 2 map on test data; (**c**) XYF-K algorithm 2 × 2 map on test data; (**d**) XYF network 4 × 4 map on training data; (**e**) XYF network 4 × 4 map on test data; (**f**) XYF-K algorithm 4 × 4 map on test data; (**g**) XYF network 8 × 8 map on training data; (**h**) XYF network 8 × 8 map on test data; (**i**) XYF-K algorithm 8 × 8 map on test data.

Figure 3a shows the 2 × 2 hexagonal map where the neurons which assume the same label have the same color, in particular, yellow for normal and red for attack. Input message data vectors X_i assume the same label of the assigned winning neurons. Figure 3a highlights that input data messages that are clearly separated by the XYF network in the training phase in two distinct neurons.

In the XYF Kohonen network, the prediction for the unknown data messages X_i is determined presenting the new object to the trained network. The network assigns the unknown message X_i to a neuron in the R_X map and classifies it as the normal or attack message according to the label of the neuron allocated in the same position as the R_Y map.

In the training process, the input data vectors are assigned to two of the four neurons, whereas two of the four neurons in the map are empty (Figure 3a). Prediction for the training set correctly classifies all the input data message vectors with an accuracy of 100% in all the datasets, since the network assigns them to the same neurons as in training process. Predictions for the test set are less accurate (Table 5) since the input data message vectors are also assigned to the neurons that were empty in the training process (Figure 3b). The network was not able to label these observations with a loss in accuracy.

Table 5. XYF Kohonen network and XYF–K algorithm accuracy.

Dataset		Training Set Accuracy (%)				Test Set Accuracy (%)		
		2 × 2	4 × 4	8 × 8		2 × 2	4 × 4	8 × 8
DoS	**XYF**	100	100	100	**XYF**	97.85	100	100
	XYF–K	100	100	100	**XYF–K**	100	100	100
Spoofing Gear	**XYF**	100	100	100	**XYF**	95.25	100	100
	XYF–K	100	100	100	**XYF–K**	100	100	100
Spoofing RPM	**XYF**	100	100	100	**XYF**	95.2	100	99.1
	XYF–K	100	100	100	**XYF–K**	100	100	100
Fuzzy	**XYF**	100	99.99	99.79	**XYF**	67.8	98.95	99.7
	XYF–K	100	99.99	99.79	**XYF–K**	98.5	99	99.75
Mixed	**XYF**	100	100	100	**XYF**	85.48	99.55	99.1
	XYF–K	100	100	100	**XYF–K**	91.79	99.55	99.1

Since some input message data vectors X_i were not classified by the XYF network, the only evaluation metric we computed was the accuracy calculated as the sum of the TP and TN over the total number of new observations presented to the network. Other metrics could return a biased representation of the network predictions.

Although the XYF network shows a high performance in predicting attack and normal messages, this lack in classifying some observations can be overcome implementing the XYF–K algorithm. After training the XYF Kohonen network, the output was subsequently processed with the K-means clustering algorithm. The neurons codebook vectors of the R_X map were used as input for the K-means algorithm, which clustered neurons in two distinct groups, defining neurons labels as normal and attack. When an unknown data message X_i is assigned by the XYF network to the winning neuron in the R_X map, it is classified as normal or attack according to the clusters defined by the K-means algorithm. Figure 3b,c highlights that the K-means algorithm also clusters empty neurons, classifying all new input data messages presented to the network. The XYF–K algorithm improves the accuracy of the XYF Kohonen network trained on a 2 × 2 neurons map in all the datasets (Table 5). In the *DoS spoofing gear* and *RPM* datasets, the XYF–K algorithm succeeded in predicting all the new data messages with an accuracy of 100%.

In order to stabilize convergence of the XYF Kohonen network, as better explained in Section 5, we expanded the 2 × 2 neurons hexagonal map introducing new interstitial nodes and obtaining first a 4 × 4 and then an 8 × 8 neurons hexagonal map. The XYF networks were trained on the expanded grids initialing the neurons codebook vectors in the R_X and R_Y maps using the neurons codebook vectors returned in output by the previously trained XYF network.

The XYF network trained using the 4 × 4 neurons map greatly improved the performance of the model in all the datasets (Figure 3e). Although the high number of periodic, sporadic and even aperiodic messages with different CAN IDs were sent at different times and with different frequencies, the network returned with an accuracy of 100% in identifying attack messages in the *DoS*, *spoofing gear* and *RPM* datasets (Table 5).

Training the XYF network with the 8 × 8 neurons map, we can see a slight improvement in accuracy only in the *fuzzy* dataset. Despite the presence of messages with 2017 different CAN IDs sent totally at random in the CAN bus, the network achieved an accuracy of 99.70%. In the *DoS* and *spoofing gear* datasets, the accuracy was the same, and in the *spoofing RPM* and *mixed* datasets, it was even worse than in the network trained using the 4 × 4 neurons map (Table 5).

The XYF–K algorithm was also implemented on the expanded neurons maps. The XYF–K algorithm slightly improved the accuracy of the XYF network in the *spoofing RPM* and *fuzzy* datasets.

In Figures 8–10, the distances between each input message data vector X_i to the corresponding winning unit returned by the XYF network were plotted for each map size. Table 6 also shows the related metrics. The boxplots highlight a clear separation in the distances for the normal and attack messages in all the datasets. This is less evident in the *fuzzy* dataset, due to the completely random structure of the dataset, and, consequently, in the *mixed* dataset. Variability in the distance is very low in the network trained with a 2 × 2 map both for normal and for attack messages with standard

deviation <0.01 in all the datasets. An increase in variability is evident when training the network using the expanded maps. These results are consistent with the accuracy of the models and confirm how clearly the XYF Kohonen network highlights a distinction between the attack and normal message data. The higher the separation in distance between the two groups, the higher the accuracy of prediction is.
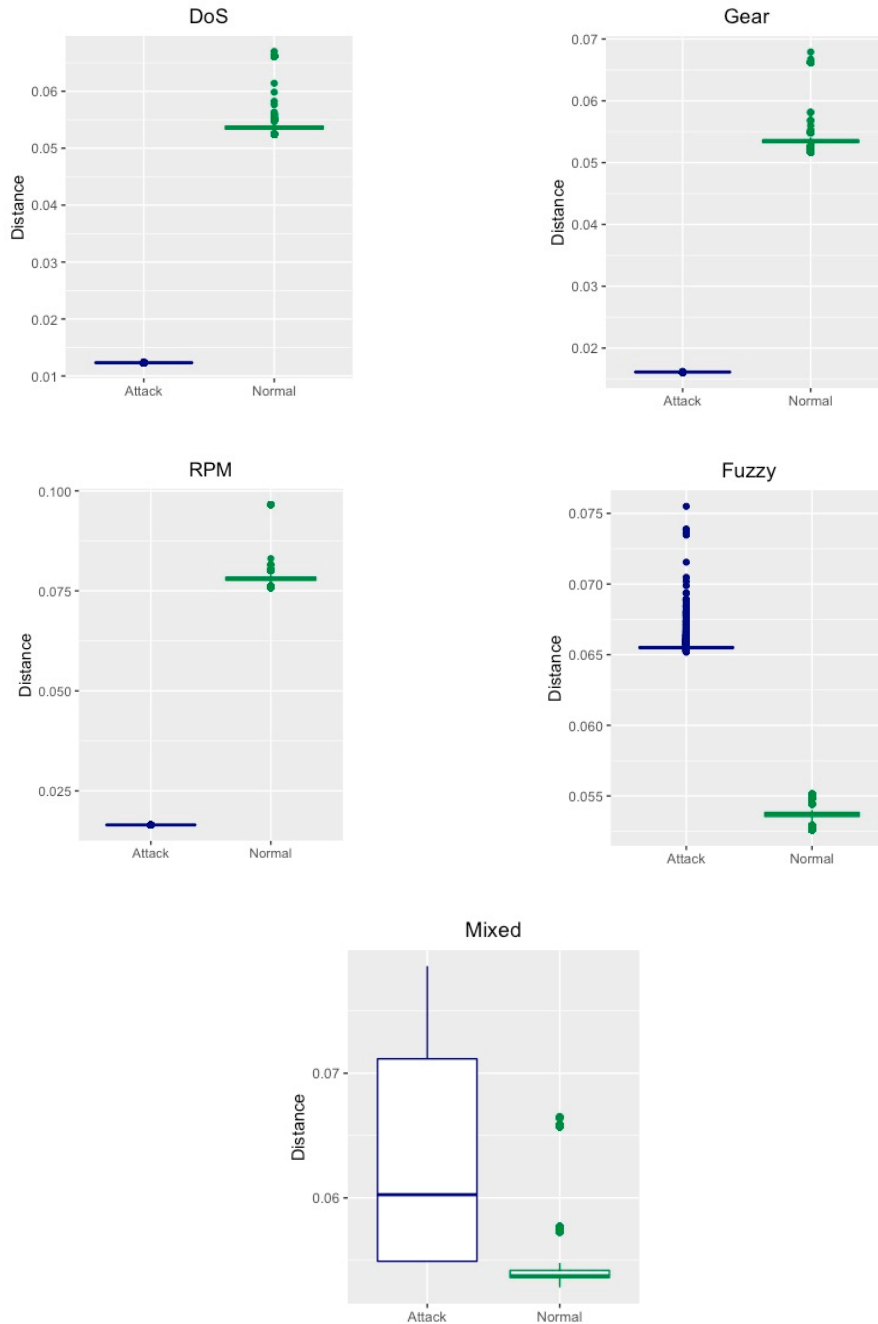

Figure 8. Distances between the input vectors and the winning neurons for XYF network on the 2 × 2 neurons map.
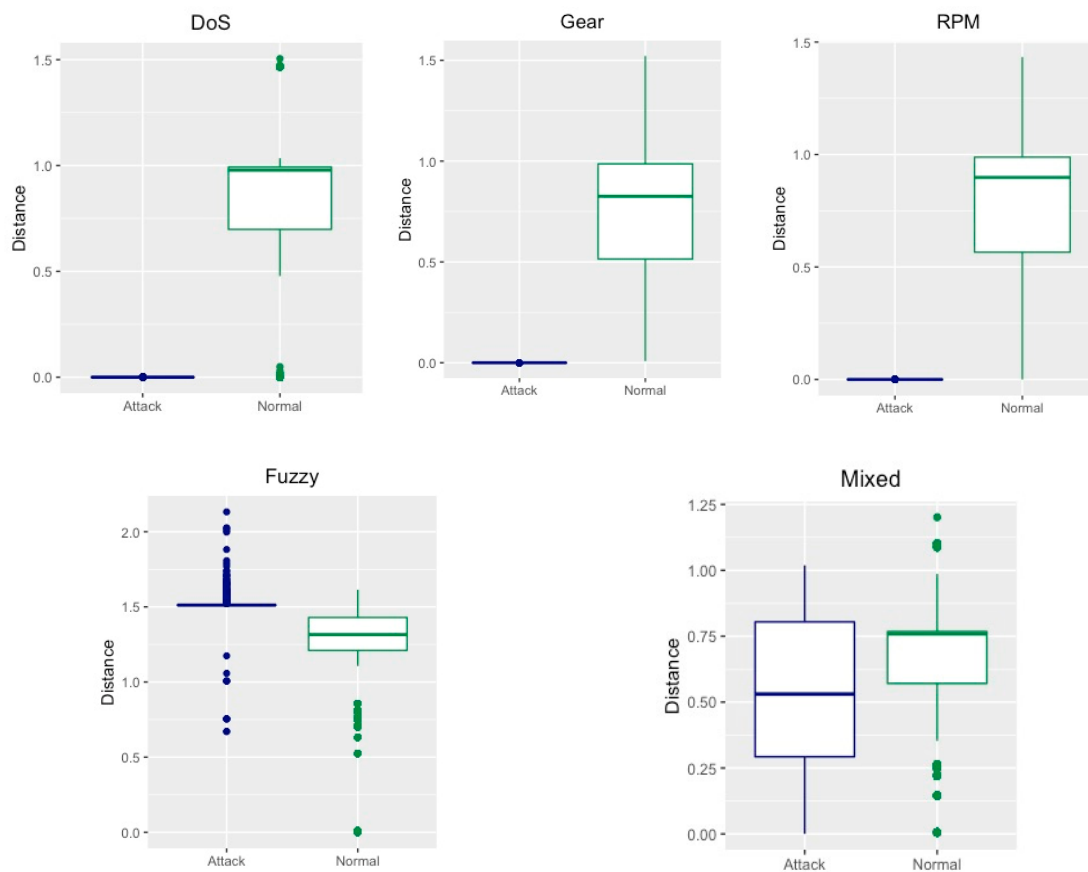
Figure 9. Distances between the input vectors and the winning neurons for the XYF network on the 4 × 4 neurons map.
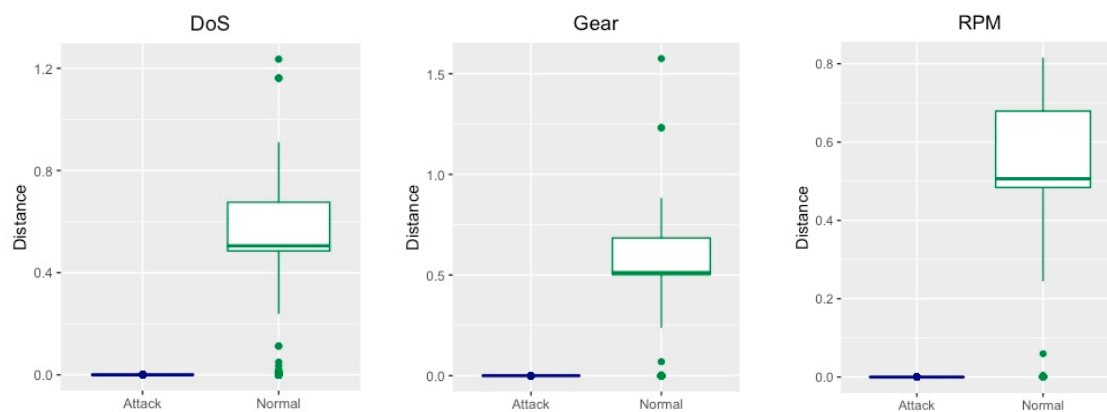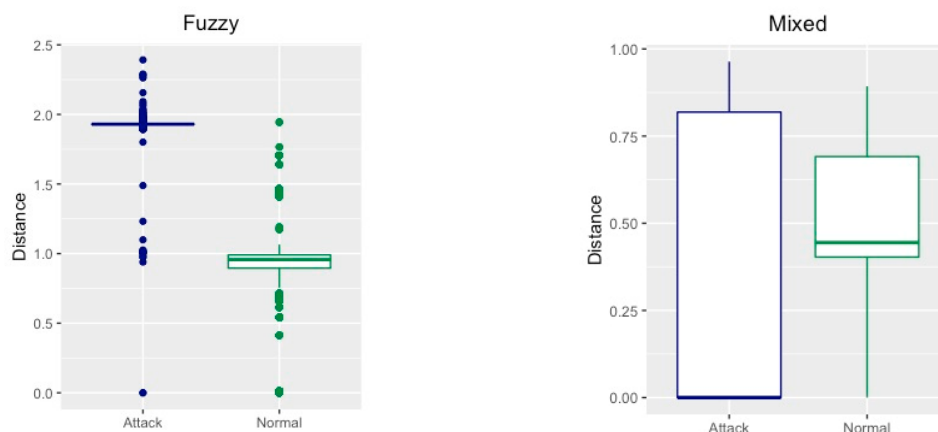
Figure 10. Distances between the input vectors and the winning neurons for the XYF network on the 8 × 8 neurons map.

Table 6. Distances between the input vectors and the winning neurons metrics in the XYF network.

Dataset	Type	Min			Median			Mean			Max			Standard Deviation		
		2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8
DoS	**Attack**	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
	Normal	0.05	0.00	0.00	0.05	0.98	0.51	0.05	0.85	0.51	0.07	1.51	1.24	0.00	0.24	0.21
Spoofing Gear	**Attack**	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00
	Normal	0.05	0.01	0.00	0.05	0.83	0.51	0.05	0.76	0.57	0.07	1.52	1.58	0.00	0.24	0.23
Spoofing RPM	**Attack**	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00
	Normal	0.08	0.00	0.00	0.08	0.90	0.51	0.08	0.81	0.52	0.10	1.43	0.82	0.00	0.24	0.22
Fuzzy	**Attack**	0.07	0.67	0.00	0.07	1.51	0.96	0.07	1.51	0.90	0.08	2.13	1.80	0.00	0.06	0.07
	Normal	0.05	0.00	0.00	0.05	1.32	1.93	0.05	1.22	1.93	0.06	1.61	2.39	0.00	0.33	0.43
Mixed	**Attack**	0.06	0.00	0.00	0.06	0.53	0.00	0.06	0.43	0.24	0.08	1.02	0.96	0.01	0.30	0.37
	Normal	0.05	0.01	0.00	0.05	0.76	0.45	0.05	0.65	0.50	0.07	1.20	0.89	0.00	0.19	0.20

For the XYF–K algorithm, we computed all the evaluation metrics as described in Section 5.4. They are shown in Table 7. Precision, recall and F1 assume values of 100% in the *DoS*, *spoofing gear* and *RPM* datasets for all map sizes. High levels of precision are also shown for the *fuzzy* dataset, 99.77% using the 8 × 8 neurons map, and for the *mixed* dataset, 99.19% with the 4 × 4 neurons map. A relevant result is given by the FNR that is 0% in the *DoS*, *spoofing gear* and *RPM* datasets, 0.90% in the *fuzzy* dataset and 1.61% in the *mixed* dataset, both with 4 × 4 and 8 × 8 neurons maps.

Table 7. Evaluation metrics for the XYF–K algorithm.

Dataset	Detection Rate (%)			Precision (%)			FPR (False Positive Rate) (%)			F1 (%)			FNR (False Negative Rate) (%)		
	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8	2 × 2	4 × 4	8 × 8
DoS	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	100.00	100.00	100.00	0.00	0.00	0.00
Spoofing Gear	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	100.00	100.00	100.00	0.00	0.00	0.00
Spoofing RPM	100.00	100.00	100.00	100.00	100.00	100.00	0.00	0.00	0.00	100.00	100.00	100.00	0.00	0.00	0.00
Fuzzy	99.10	99.10	99.10	94.42	96.49	99.77	0.02	0.01	0.00	96.70	97.78	99.44	0.90	0.90	0.90
Mixed	75.87	98.39	98.39	79.27	99.19	96.83	0.05	0.00	0.01	77.53	98.79	97.61	24.13	1.61	1.61

Table 8 shows a comparison between the XYF–K algorithm and other IDS approaches proposed to detect attack messages sent on the CAN bus. We compared the proposed method in terms of the system's accuracy with all the studies that, at the state of the art, tested different IDS approaches on the same dataset (**RQ2**).

Song et al. proposed an anomaly-based IDS using a supervised deep convolutional neural network (DCNN) [30]. Experimental results showed that their approach reached a higher performance compared to other machine learning algorithms, in particular, long short-term memory, artificial neural network, support vector machine, k-nearest neighbors, naive Bayes, and decision trees. Olufowobi et al. suggested a specification-based IDS using a response time analysis of the CAN bus. They restricted the analysis only checking periodic and sporadic messages [38]. Seo et al. proposed an anomaly-based IDS using a deep-learning model, in particular, generative adversarial nets (GAN) [27].

Results (Table 8) show how the XYF–K algorithm presents superior performance compared to all the other methods for all the datasets. Only the DCNN proposed by [30] reveals a slightly better accuracy for the fuzzy dataset, in particular, the authors claim an accuracy of 99.82% while in our study it is equal to 99.75%. Overall, we can affirm that the XYF–K algorithm improves the performance than those of the state of the art.

Table 8. Comparison between the XYF–K algorithm and the methods previously studied.

Method	Detection Strategy	Model	Accuracy (%)			
			DoS	*Spoofing Gear*	*Spoofing RPM*	*Fuzzy*
Song et al. [30]	Anomaly based	Deep convolutional neural network (DCNN)	99.97	99.95	99.00	99.82
Olufowobi et al. [30]	Specification based	Time interval analysis	98.08	82.62	80.33	87.82
Seo et al. [27]	Anomaly based	Generative adversarial network (GAN)	99.90	99.80	99.60	98.70
Proposed method	Anomaly based	Self organizing map (SOM)	100.00	100.00	100.00	99.75

Figures 11–13 show the training process of the XYF Kohonen network for the different map sizes on all the datasets. It is evident that the networks trained on 2 × 2 grids are not stable in convergence, whereas networks trained on expanded 4 × 4 and 8 × 8 neurons maps have more stability in convergence. The networks trained on both the 4 × 4 and 8 × 8 neurons grids start to stabilize after 450 iterations.
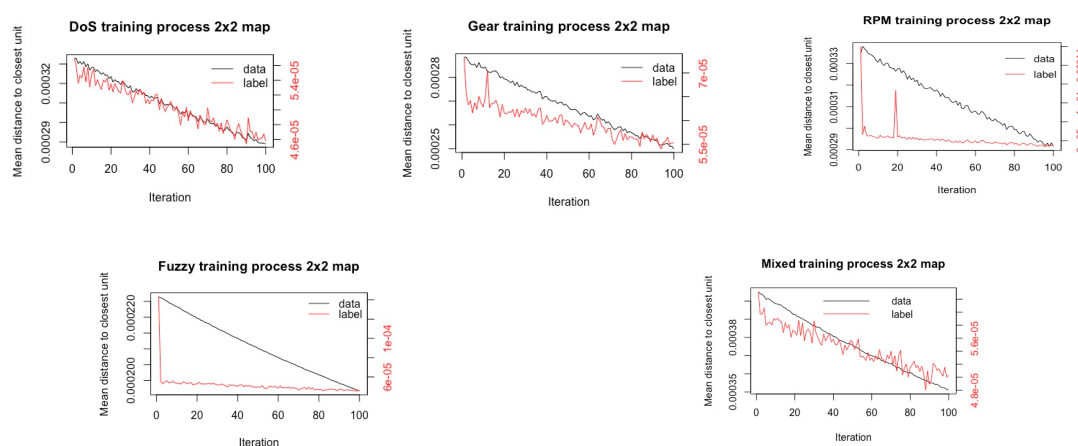

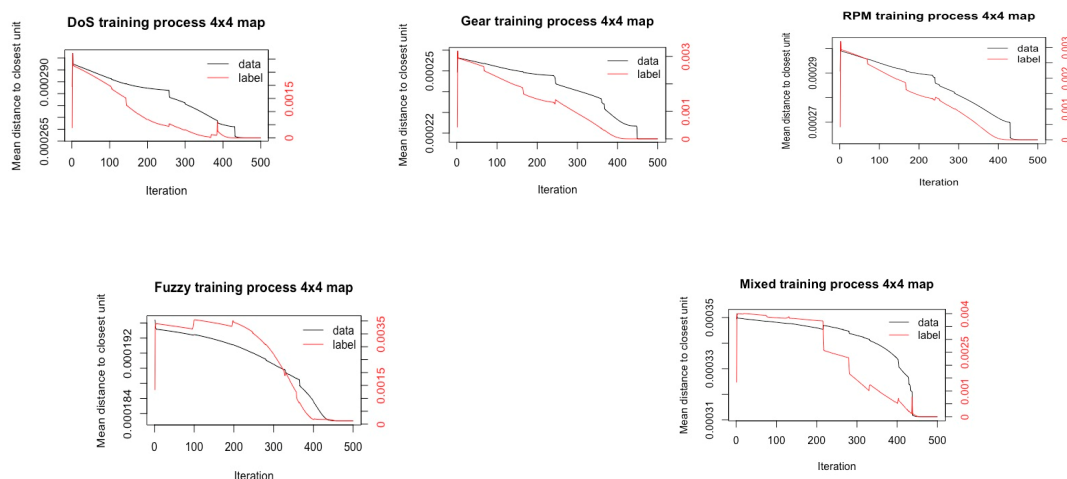**Figure 11.** XYF Kohonen network training process for the 2 × 2 maps.

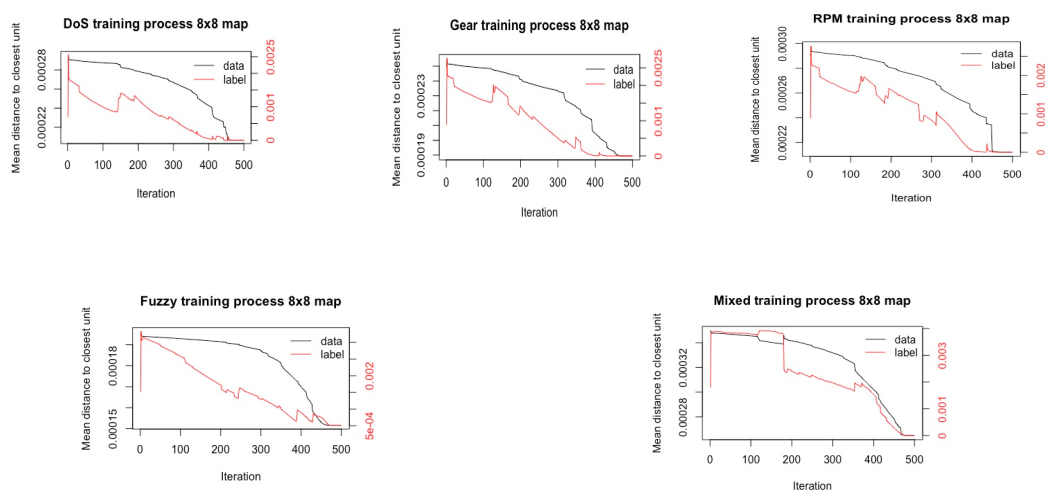**Figure 12.** XYF Kohonen network training process for the 4 × 4 maps.

Figure 13. XYF Kohonen network training process for the 8 × 8 maps.

7. Conclusions

The increasing diffusion of the connected devices in modern vehicles implies a lack in security of the in-vehicle systems such as the CAN bus. Since the CAN bus protocol does not provide security systems against cyber and physical attacks, it is desirable to have a system to detect any kind of attack.

In the present work, we implemented a supervised distance-based intrusion-detection system with the aim of identifying the attack messages injected on a CAN bus. We performed two different methods, the XYF Kohonen network and the XYF–K algorithm. We tested both the models on four open source car hacking datasets containing different types of attack, namely, *DoS*, *spoofing gear*, *spoofing RPM*, and *fuzzy*. First, single datasets were trained one by one, then the four datasets were combined together. The XYF Kohonen network showed a great performance in identifying attack messages injected on the CAN bus in all the datasets. The XYF–K algorithm, which processed the output of the XYF network using a K-means algorithm, slightly improved the performance of the network. Despite the high number of messages with different CAN ID identifiers sent at different times and with different frequencies, the experimental results showed that both the models completely succeeded in the prevision of all types of attack messages in *DoS*, *spoofing gear* and *spoofing RPM*. Moreover, both models achieved high levels of accuracy even in the *fuzzy* dataset, despite the presence of 2017 different CAN ID identifiers sent completely at random, and in the *mixed* dataset.

The limit of the model lies in the fact that the SOM network can be time-consuming since it performs a computation at every step of the algorithm. Nevertheless, the architecture of the network enables to run the algorithm in parallel computations. Thus, the diffusion of multi core computational architectures overcomes limitations due to the complexity of the network architecture enhancing the interest of researchers in the application of the Kohonen SOM networks in security issues [15].

In this paper, we tested the proposed method on a limited number of attacks, due to the availability of the open source dataset we analyzed. In future works, we will extend the analysis to other types of attacks. Moreover, we will investigate the development of the Kohonen SOM network as anomaly detector on a CAN bus, also comparing the supervised and unsupervised learning.

Author Contributions: Conceptualization, D.C.; data curation, V.S.B., A.N. and M.S.; formal analysis, A.N.; investigation, V.S.B. and D.C.; methodology, D.C. and A.N.; project administration, D.C.; software, V.S.B., A.N. and M.S.; supervision, M.S.; validation, V.S.B. and D.C.; visualization, V.S.B. and A.N.; writing—original draft, V.S.B. and A.N.; writing—review and editing, D.C. and M.S.

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

References

1. Karoń, G.; Żochowska, R.; Problems of Quality of Public Transportation Systems in Smart Cities—Smoothness and Disruptions in Urban Traffic. In *Modelling of the Interaction of the Different Vehicles and Various Transport Modes*; Springer: Cham, Switzerland, 2020; pp. 383–414.

2. Soczówka, P.; Żochowska, R.; Karoń, G. Method of the Analysis of the Connectivity of Road and Street Network in Terms of Division of the City Area. *Computation* **2020**, *8*, 54.

3. Barletta, V.S.; Caivano, D.; Dimauro, G.; Nannavecchia, A.; Scalera, M. Managing a Smart City Integrated Model through Smart Program Management. *Appl. Sci.* **2020**, *10*, 1–23, doi:10.3390/app10020714.

4. Baldassarre, M.T.; Barletta, V.S.; Caivano, D. Smart Program Management in a Smart City. In Proceedings of the 2018 110th AEIT International Annual Conference (AEIT), Bari, Italy, 3–5 October 2018, doi:10.23919/AEIT.2018.8577379.

5. Caivano, D. Continuous software process improvement through statistical process control. In Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR), Manchester, UK, 21–23 March 2005; pp. 288–293.

6. Baldassarre, T.; Boffoli, N.; Caivano, D.; Visaggio, G. Managing Software Process Improvement (SPI) through Statistical Process Control (SPC). In: Bomarius F., Iida H. (eds) International conference on Product Focused Software Process Improvement; Springer, Berlin, Germany, 2004; Volume 3009, pp. 30–46, doi:10.1007/978-3-540-24659-6_3

7. Dimauro, G.; Girardi, F.; Caivano, D.; Colizzi, L. Personal Health E-Record—Toward an Enabling Ambient Assisted Living Technology for Communication and Information Sharing between Patients and Care Providers. In *Lecture Notes in Electrical Engineering*; Ambient Assisted Living. ForItAAL 2018; Leone, A., Caroppo, A., Rescio, G., Diraco, G., Siciliano, P., Eds.; Springer: Cham, Switzerland, 2019; Volume 544, doi:10.1007/978-3-030-05921-7_39.

8. Baldassarre, M.T.; Barletta, V.S.; Caivano, D.; Scalera, M.; Integrating security and privacy in software development. *Softw. Qual. J.* **2020**, 1–32, doi:10.1007/s11219-020-09501-6.

9. Baldassarre, M.T.; Barletta, V.S.; Caivano, D.; Scalera, M. Privacy Oriented Software Development. In: Piattini M., Rupino da Cunha P., García Rodríguez de Guzmán I., Pérez-Castillo R. (eds) International Conference on the Quality of Information and Communications Technology; Springer: Cham, Switzerland 2019; Volume 1010; pp. 18–32, doi:10.1007/978-3-030-29238-6_2.

10. Baldassarre, M.T.; Barletta, V.S.; Caivano, D.; Raguseo, D.; Scalera, M. Teaching cyber security: The hack-space integrated model. In Proceedings of the CEUR Workshop Proceedings, **2019**; Volume 2315. ITASEC, Pisa, Italy, 13-15 February 2019.

11. Woo, S.; Jo, H.J.; Lee, D.H. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 993–1006, doi:10.1109/TITS.2014.2351612.

12. Kleberger, P.; Olovsson, T.; Jonsson, E. Security aspects of the in-vehicle network in the connected car. *IEEE Intell. Veh. Symp. Proc.* **2011**, 528–533, doi:10.1109/IVS.2011.5940525.

13. Sommer, F.; Dürrwang, J. Survey and Classification of Automotive Security Attacks. *Information* **2019**, *10*, 148, doi:10.3390/info10040148.

14. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of automotive controller area network intrusion detection systems. *IEEE Design Test* **2019**, *36*, 48–55.

15. Yao, X.Q.; Tang, G.; Hu, X. Method for recognizing mechanical status of container crane motor based on SOM neural network. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing Ltd.: Bristol, UK, 2018; Volume 435, p. 12009.

16. Wu, Y.; Yan, P.F. A study on structural adapting self-organizing neural network. *Acta Electron. Sin.* **1999**, *27*, 56–59.

17. Wan, Q.; Wang, C.; Feng, Z.Y.; Ye, J.F. Review of K-means clustering algorithm. *Electron. Design Eng.* **2012**, 20, 21–24.

18. Feyereisl, J.; Aickelin, U. Self-organizing maps in computer security. In *Computer Security: Intrusion, Detection and Prevention*; Editors: Ronald D. Hopkins et al; Nova Science Publishers Inc., New York. **2009**, pp. 1–30.

19. Barletta, V.S.; Caivano, D.; Nannavecchia, A.; Scalera, M. Intrusion Detection for In-Vehicle Communication Networks: An Unsupervised Kohonen SOM Approach. *Future Internet* **2020**, *12*, 119, doi:10.3390/fi12070119.

20. Car-Hacking Dataset for the Intrusion Detection. Available online: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset. (accessed on 27 November 2019).

21. El-Rewini, Z.; Sadatsharan, K.; Selvaraj, D.F.; Plathottam, S.J.; Ranganathan, P. Cybersecurity challenges in vehicular communications. *Veh. Commun.* **2020**, *23*, 100214, doi:10.1016/j.vehcom.2019.100214.

22. Ueda, H.; Kurachi, R.; Takada, H.; Mizutani, T.; Inoue, M.; Horihata, S. Security authentication system for in-vehicle network. *SEI Tech. Rev.* **2015**, *81*, 5–9.

23. Liu, J.; Sun, W.; Shi, Y. In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Netw.* **2017**, *31*, 50–58, doi:10.1109/MNET.2017.1600257.

24. Hasbullah, H.; Soomro, I.A.; Manan, J. Denial of Service (DOS) Attack and Its Possible Solutions in VANET. *Int. J. Electron. Commun. Eng.* **2010**, *4*, 813–817.

25. Takada, M.; Osada, Y.; Morii, M. Counter Attack Against the Bus-Off Attack on CAN. In Proceedings of the 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), Kobe, Japan, 1–2 August 2019; pp. 96–102, doi:10.1109/AsiaJCIS.2019.00004.

26. Lokman S.F.; Othman A.T.; Musa S.; Abu Bakar M.H. Deep Contractive Autoencoder-Based Anomaly Detection for In-Vehicle Controller Area Network (CAN). In *Progress in Engineering Technology*; Advanced Structured Materials; Abu Bakar, M., Mohamad Sidik, M., Öchsner, A., Eds.; Springer: Cham, Switzerland, 2019; Volume 119, doi:10.1007/978-3-030-28505-0_16.

27. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In Proceedings of the 16th Annual Conference on Privacy, Security and Trust, Belfast, UK, 28–30 August 2018, pp. 1–6, doi:10.1109/PST.2018.8514157.

28. Martínez Torres, J.; Iglesias Comesaña, C.; García-Nieto, P.J. Review: Machine learning techniques applied to cybersecurity. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 2823–2836, doi:10.1007/s13042-018-00906-1.

29. 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**, *2019*, 184, doi:10.1186/s13638-019-1484-3.

30. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, doi:10.1016/j.vehcom.2019.100198.

31. Kang, M.; Kang, J. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLoS ONE* **2016**, *11*, e0155781.

32. Zhou, A.; Li, Z.; Shen, Y. Anomaly detection of CAN bus messages using a deep neural network for autonomous vehicles. *Appl. Sci.* **2019**, *9*, 3174, doi:10.3390/app9153174.

33. Hanselmann, M.; Strauss, T.; Dormann, K.; Ulmer, H. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access* **2020**, *8*, 58194–58205, doi:10.1109/ACCESS.2020.2982544.

34. Chan, A.T.S.; Shiu, A.; Cao, J.; Leong, H.V. Reactive Web policing based on self-organizing maps. In Proceedings of the IEEE Region 10 International Conference on Electrical and Electronic Technology, Singapore, 19–22 August 2001; pp. 160–164, doi:10.1109/tencon.2001.949572.

35. Oh, H.; Lim, J.; Chae, K.; Nah, J. Home gateway with automated real-time intrusion detection for secure home networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2006**, Vol. 3983 LNCS, pp. 440–447, doi:10.1007/11751632_48.

36. Quah, J.T.S.; Sriganesh, M. Real-time credit card fraud detection using computational intelligence. *Expert Syst. Appl.* **2008**, *35*, 1721–1732, doi:10.1016/j.eswa.2007.08.093.

37. Tan, L.; Li, C.; Xia, J.; Cao, J. Application of self-organizing feature map neural network based on K-means clustering in network intrusion detection. *Comput. Mater. Contin.* **2019**, *61*, 275–288, doi:10.32604/cmc.2019.03735.

38. Olufowobi, H.; Young, C.; Zambreno, J.; Bloom, G. SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 1484–1494, doi:10.1109/TVT.2019.2961344.

39. Barbieri, N. Fuel prices and the invention crowding out effect: Releasing the automotive industry from its dependence on fossil fuel. *Technol. Forecast. Soc. Change* **2016**, *111*, 222–234, doi:10.1016/j.techfore.2016.07.002.

40. Akinduko, A.A.; Mirkes, E.M. Initialization of self-organizing maps: Principal components versus random initialization. A case study. *arXiv* **2012**, 3–20, arXiv:1210.5873.

41. Ciaburro G.; Venkateswaran, B. *Neural Networks with R: Smart Models Using CNN, RNN, Deep Learning, and Artificial Intelligence Principles*; Packt Publishing Ltd.: Birmingham, UK, 2017.

42. Shamsuddin, S.M.; Zainal, A.; Mohd Yusof, N. Multilevel Kohonen Network Learning for Clustering Problems. *J. Inf. Commun. Technol. (JICT)* **2008**, *7*, 1–25.

43. Wehrens, R.; Buydens, L.M.C. Self- and super-organizing maps in R: The kohonen package. *J. Stat. Softw.* **2007**, *21*, 1–19, doi:10.18637/jss.v021.i05.

44. Kohonen, T. *Self-Organizing Maps*; Springer: Berlin Heidelberg, 2001; Volume 30, doi:10.1007/978-3-642-56927-2.

45. Wehrens, R.; Kruisselbrink, J. Flexible self-organizing maps in kohonen 3.0. *J. Stat. Softw.* **2018**, *87*, doi:10.18637/jss.v087.i07.

46. Vasighi, M.; Kompany-Zareh, M. Classification ability of self organizing maps in comparison with other classification methods. *Commun. Math. Comput. Chem.* **2013**, *70*, 29–44.

47. Dimauro, G.; Guarini, A.; Caivano, D.; Girardi, F.; Pasciolla C.; Iacobazzi, A. Detecting Clinical Signs of Anaemia From Digital Images of the Palpebral Conjunctiva. *IEEE Access* **2019**, *7*, 113488–113498, doi:10.1109/ACCESS.2019.2932274.

48. Kohonen, T. Essentials of the self-organizing map. *Neural Netw.* **2013**, *37*, 52–65, doi:10.1016/j.neunet.2012.09.018.

49. Melssen, W.; Wehrens, R.; Buydens, L. Supervised Kohonen networks for classification problems. *Chemom. Intell. Lab. Syst.* **2006**, *83*, 99–113, doi:10.1016/j.chemolab.2006.02.003.

50. Yedla, M.; Pathakota, S.R.; Srinivasa, T.M. Enhancing K-means clustering algorithm with improved initial center. *Int. J. Comput. Sci. Inf. Technol.* **2020**, *1*, 121–125.

51. Wang, H.B.; Yang, H.L.; Xu, Z.J.; Yuan, Z. A clustering algorithm use SOM and K-Means in Intrusion Detection. In Proceedings of the International Conference on E-Business and E-Government (ICEE), Guangzhou, China, 7–9 May 2010; pp. 1281–1284, doi:10.1109/ICEE.2010.327.

52. Ardimento P., Baldassarre M.T., Caivano D., Visaggio G. Multiview Framework for Goal Oriented Measurement Plan Design. In: Bomarius F., Iida H. (eds) Product Focused Software Process Improvement. PROFES 2004. Lecture Notes in Computer Science, volume 3009. Springer, Berlin, Heidelberg, doi:10.1007/978-3-540-24659-6_12.

53. Baldassarre, M.T.; Caivano, D.; Visaggio, G.; Empirical studies for innovation dissemination: ten years of experience. In Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE '13). Association for Computing Machinery, New York, NY, USA, April 2013; 144–152. doi:10.1145/2460999.2461020.

54. Dimauro, G.; Altomare, N.; Scalera, M. PQMET: A digital image quality metric based on human visual system. In Proceedings of the 2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA), Paris, France, 14–17 October 2014; pp. 1–6, doi:10.1109/IPTA.2014.7002002.

55. Kumar, G.; Evaluation Metrics dor Intrusion Detection System—A Study. *Int. J. Comput. Sci. Mob. Appl.* **2014**, *2*, 11–17. ISSN: 2321-8363.