future internet

Article

Intrusion Detection for In-Vehicle Communication Networks: An Unsupervised Kohonen SOM Approach

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

[1] Department of Informatics, University of Bari, Via E. Orabona 4, 70125 Bari, 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, 70010 Casamassima (BA), Italy

* Correspondence: nannavecchia@lum.it

Received: 15 June 2020; Accepted: 11 July 2020; Published: 14 July 2020

Abstract: The diffusion of embedded and portable communication devices on modern vehicles entails new security risks since in-vehicle communication protocols are still insecure and vulnerable to attacks. Increasing interest is being given to the implementation of automotive cybersecurity systems. In this work we propose an efficient and high-performing intrusion detection system based on an unsupervised Kohonen Self-Organizing Map (SOM) network, to identify attack messages sent on a Controller Area Network (CAN) bus. The SOM network found a wide range of applications in intrusion detection because of its features of high detection rate, short training time, and high versatility. We propose to extend the SOM network to intrusion detection on in-vehicle CAN buses. Many hybrid approaches were proposed to combine the SOM network with other clustering methods, such as the k-means algorithm, in order to improve the accuracy of the model. We introduced a novel distance-based procedure to integrate the SOM network with the K-means algorithm and compared it with the traditional procedure. The models were tested on a car hacking dataset concerning traffic data messages sent on a CAN bus, characterized by a large volume of traffic with a low number of features and highly imbalanced data distribution. The experimentation showed that the proposed method greatly improved detection accuracy over the traditional approach.

Keywords: intrusion detection systems; unsupervised learning; self-organizing maps; CAN bus; Kohonen SOM network; cyber–physical systems; security; vehicle safety; cyber-attacks

1. Introduction

The automotive sector has been undergoing a radical transformation in recent years. Vehicles' cyber–physical systems are partially or totally controlled by software run by electronic devices increasingly interconnected with the outside world through networks of various types [1]. A number of initiatives concerning smart mobility [2] and autonomous driving are, indeed, experiencing increasing development in urban areas and smart city contexts [3,4].

Automatic systems to maintain the lane, cruising speed, and movement in the queue; to park automatically; and to check the state of attention and sobriety of the driver [5], are just some of the advantages. At the same time, these advantages have significantly increased the attack surface area. There are several access points [6] that an attacker can use to try to compromise the security of the vehicle: connections to smartphones; USB inputs; the mobile network to receive information, transmit data, and make calls to external services [7,8]; Wi-Fi connections that can be used to connect other mobile devices on board the vehicle. Those are just some of the known channels [9] and in this scenario, continuous software improvements [10,11] become necessary in order to detect and respond in time to possible attacks. Proper methods and tools capable of managing the complexity [12] can guarantee not only the protection of the vehicle, but also and above all, that of the people.

In this research, we take into consideration the vehicle's internal network, the Controller Area Network (CAN), as it allows safety-critical electronic control units (ECUs) that are attached to sufficiently broadcast information in the form of CAN packets between them and other connected busses through several gateways [13].

The CAN bus system presents several critical vulnerabilities [14]. For example, receiving nodes are unable to verify whether the received packet is legitimate or not since the origin of the packets is not provided. Consequently, ECUs can be used by attackers to falsify and send fake CAN packets. This makes a CAN bus system insecure and poorly equipped in order to identify which nodes launched the attacks.

In view of these considerations, security systems to protect the CAN bus became an urgent need [15]. Intrusion detection techniques traditionally used in network security cannot be implemented in the automotive domain. Since network-based attacks are relatively new in the automotive field, new challenges arise in the development of efficient and adaptable systems for securing automotive networks [16]. Since the CAN protocol can be frequently modified, a Machine Learning approach could be the proper way to implement a detection method which, learning by examples, is able to adapt to any change in the protocol. Most of the intrusion detection systems (IDSs) based on Machine Learning proposed in the literature are deployed in a supervised manner. This requires data to be completely labelled, even if this can be unfeasible considering the high volume of data generated by a real time CAN in milliseconds [13]. Thus, an anomaly-based detection system implemented using an unsupervised Machine Learning approach is more desirable and convenient.

In the present work, we introduce a distance-based intrusion detection system aimed at identifying attacks sent on the CAN bus. The proposed system was based on an unsupervised Kohonen Self-Organizing Map (SOM) network, an Artificial Neural Network that can be trained both through supervised and unsupervised learning. The algorithm maps a high-dimensional data space to a low-dimensional one, preserving the topological properties of input data. It is a power classifier able to separate normal from anomalous data while preserving the topological relationship between the features with no need for labels. Due to the powerful capabilities in clustering and visualization of complex, highly dimensional data, the SOM network extensively evolved, thereby finding many applications in intrusion detection [17]. The algorithm showed high performance as an anomaly detector in real-time systems, and compared to other intrusion detection techniques, revealed better performance by showing a shorter training time and higher detection efficiency [18–20]. The SOM network is finding applications in new areas of security, never approached in a similar way before, such as anomaly-based detection [21]. In the light of these, we propose to test the effectiveness and the efficiency of an unsupervised Kohonen SOM network in the automotive domain, since, to the best of our knowledge, it was never tested in said area before.

Many hybrid methods based on the integration of the Kohonen SOM network with other clustering methods were proposed in order to improve detection accuracy and reduce false alarm rates. One of the most common methods is based on the combination of the Kohonen SOM network with the k-means algorithm [22,23]. Intrusion detection networks were usually tested on the well-known KDD99 dataset and on its refined version, the NSL-KDD dataset [24]. These datasets have large numbers of features and experiments. References [25–28] showed that detection accuracy achieved its high performance when including all the features in the analysis. The richer the feature space is, the higher the detection rate achieved [17].

The goal of the present research is to evaluate the performance of an anomaly-based intrusion detection system using an unsupervised Kohonen SOM neural network for the identification of attack messages sent on the CAN bus. We propose a novel distance-based procedure to integrate

Kohonen SOM network and k-means algorithm, which greatly improved accuracy in detecting attack messages compared to the traditional procedure.

The proposed method and the traditional procedure based on the combination of the Kohonen SOM network and k-means algorithm, were tested on open source data concerning traffic messages sent on a CAN bus 2.0B with a very complex structure, characterized by large volume of traffic with low number of features and a highly imbalanced data distribution. The dataset contains more than 2000 different kinds of messages sent totally at random on the CAN bus and all included in the analysis. Despite the complex structure of the dataset, the proposed method showed high detection accuracy with a low false negative rate.

The main contributions of the paper are:

- An anomaly-based IDS implemented using unsupervised learning to identify intrusions on an in-vehicle communication network, in particular, a CAN bus. At the state of the art level, this is the first work which tests an unsupervised Kohonen SOM network as an anomaly detector in the automotive domain.
- A novel distance-based procedure to integrate Kohonen SOM network and k-means algorithm, which greatly improves accuracy in detecting attack messages compared to the traditional procedure. Moreover, the proposed method significantly reduces false negative rate, which assumes a great importance in attack detection for in-vehicle CAN buses. Its value should be very low to ensure the safety of the vehicle.
- The proposed method was tested on real car-hacking data, including *DoS*, *spoofing the drive gear*, *spoofing the RPM gauge*, and *fuzzy* attacks. Data were obtained by logging CAN traffic via the OBD-II port from a real vehicle while message injection attacks were being performed. The performance of the proposed method was shown via computing evaluation metrics.
- The proposed method was performed with remarkable results both using single datasets only containing one type of attack and also merging all types of attack into a unique dataset.
- Most of the studies in the literature propose IDSs only able to detect periodic attacks but not aperiodic violations. The proposed method reveals remarkable performance in detecting both periodic and aperiodic intrusions.

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

2. Related Works

Recent works showed that data generated by connected vehicles can be a great resource for the development of next generation cybersecurity solutions [29]. In [30,31] the authors highlight the trend of increasing research interest in applications of Machine Learning (ML) and Deep Learning (DL) in cybersecurity for the automotive industry.

The learning types and their applicability in the automotive industry are: *supervised ML models* which deal with labelled data concerning automotive used to train the ML classification model; *unsupervised/self-supervised ML models* that create clusters from various vehicle data streams with no need for labelled data can be further analyzed to detect abnormal behavior; *reinforcement learning models*, although less mature than the first two, provide a means to develop autonomous cybersecurity solutions that can take human-defined meta-goals as input and make decisions to achieve that goal [29].

A cybersecurity solution need not to be limited using a single architecture [32] or a single model [33], and in accordance with that, our research investigates the cybersecurity solutions which propose a distance-based intrusion detection system based on an unsupervised Kohonen SOM network. Therefore, our goal is to test the SOM network in order to identify attacks within the CAN bus.

In the literature there are different ML models applied to automotive cybersecurity solutions (Table 1), such as the Bayesian network, to determine whether the vehicle is under attack, but also whether the attack has originated from the cyber or the physical domain [34]; the Deep Neural Network (DNN) to train in-vehicle network packets exchanged between ECUs to extract low-dimensional features, and it is used for discriminating normal and hacking packets [35]; the long short-term memory neural network to detect CAN bus attacks [36]; Convolutional Neural Networks (CNNs) to classify malware samples [37]; and Generative Adversarial Networks to generate the adversarial attacks, which can deceive and evade the intrusion detection system [38].

Each of them allows providing a solution for a specific class within the vehicle (network security, VANET situational awareness, vehicle intelligence, and others [29]), but there is no trace in the literature of a SOM network application in automotive context. The Kohonen SOM network is a popular non-linear model of unsupervised neural network for the solution of dimensionality reduction problems [39] and it found mostly in applications concerning security issues [17] because of its features of high detection rate, short training time, and high versatility [22].

Starting from the results obtained in the application of the Kohonen SOM network, we extended this model in the automotive domain. We propose an intrusion detection system to identify attack messages sent on the CAN bus based on an unsupervised Kohonen SOM network. In many studies, the SOM network was integrated with other clustering methods in order to improve the efficiency of the model [17]. Wang et all. in [23] combined the unsupervised SOM network with the k-means algorithm using two different approaches and tested both methods on the KDD CUP-99 dataset, commonly used to test intrusion detection system. Their methods showed good stability of efficiency and clustering accuracy. Tan et all. in [22] also proposed an intrusion detection method based on the integration of the unsupervised SOM network with the k-means algorithm and tested their model on the NSL-KDD dataset, a refined version of the KDD CUP-99. Both the datasets are characterized by a high number of features, equal to 41, for each connection record. The proposed method relatively improved the accuracy of network intrusion and significantly reduced the number of clustering iterations than the SOM network.

We applied and evaluated the performance of the unsupervised Kohonen SOM network as an intrusion detection system on an in-vehicle communication network, in particular, on a CAN bus. We present an intrusion detection system based on a novel distance-based procedure for the integration of the Kohonen SOM network with the k-means algorithm and compare it with the traditional procedure. Performance of classification was statistically tested for the two methods using open source car hacking data concerning the traffic of messages sent on CAN bus 2.0B and consisting of four datasets, each containing a different type of attack. The same dataset was used to test the intrusion detection system proposed by [40] and [41]. In the analysis, they individually considered each dataset containing a unique type of attack, whereas we tested the models first on a single dataset, and then by merging the four datasets into one containing all the four different types of attack. The structure of the data was very complex, containing a large volume of traffic with a low number of features, equal to 4, and a highly imbalanced data distribution.

The experimentation showed a great improvement in the accuracy of attack message detection and a significantly reduced false negative rate compared to the traditional procedure.

Table 1. Machine learning solution in automotive.

References	ML model	Solution
[34]	Bayesan	Intrusion detection
[35]	DNN	Intrusion detection
[36]	LSTM	Intrusion Detection
[37]	CNN	Malware Classification
[38]	GAN	Attack simulation
[40]	DCNN	Intrusion Detection

3. Control Area Network (CAN)

A Controller Area Network, or CAN, is the most commonly used network for control in automotive and manufacturing applications [42]. The CAN interconnects a network of nodes and it is a serial, multimaster, multicast protocol, so when the bus is free any node can send a message, and all nodes may receive and act on the message. When a node begins to transmit messages, it prioritizes the messages. This allows you to transmit until the bus becomes inactive or until it is replaced by a node with a higher priority message.

There are four types of CAN messages: data frame (CAN 2.0A and CAN2.0B), which is the standard CAN message broadcasting data from the transmitter to the other nodes on the bus; remote frame, a message that is broadcast by a transmitter to request data from a specific node; an error frame may be transmitted by any node that detects a bus error; overload frames are used to introduce additional delay between data or remote frames . In this research, CAN 2.0B data frame [40] was taken into consideration (Table 2). The difference between a CAN 2.0A and a CAN 2.0B message is that CAN 2.0B supports both 11 bit (standard) and 29 bit (extended) identifiers.

Table 2. Controller Area Network (CAN) bus 2.0A and CAN bus 2.0B structure.

Field	CAN Message	Bits (Length)	Description
SOF	CAN 2.0A	1	The Start of Frame (SOF) indicates the start of a new message and the single bit must be dominant as it is used for synchronizing all nodes
	CAN 2.0B	1	
Identifier	CAN 2.0A	11	The identifier establishes the priority of the message: the lower the value, the higher its priority
	CAN 2.0B	11	This identifier is the first part, which is used in both the standard and extended frames
Identifier extended	CAN 2.0A	-	-
	CAN 2.0B	18	In CAN 2.0B Extended, the identifier is comprised of 11 bits Base ID and 18 bits Extended ID
RTR	CAN 2.0A	1	The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform [43]
	CAN 2.0B	1	
SRR	CAN 2.0A	-	-
	CAN 2.0B	1	The Substitute Remote Request (SRR) must be recessive and used in the extended frame
IDE	CAN 2.0A	-	-
	CAN 2.0B	1	The Identifier Extension (IDE) must be recessive for extended format and dominant for standard format
Reserved r0	CAN 2.0A	-	-
	CAN 2.0B	1	Reversed bit must be dominant for standard format
Reserved r0, r1	CAN 2.0A	2	Bits must be dominant
	CAN 2.0B	2	Bits must be recessive for extended format
DLC	CAN 2.0A	4	The data length code (DLC) contains the number of data bytes
	CAN 2.0B	4	
Data Field	CAN 2.0A	64	The actual payload data which can be up to 64 bits

	CAN 2.0B	64	
CRC	CAN 2.0A	15	Cyclic Redundancy Check (CRC) contains the checksum of the previous data for error detection
	CAN 2.0B	15	
CRC Delimiter	CAN 2.0A	1	The single bit must be recessive
	CAN 2.0B	1	
ACK	CAN 2.0A	1	Acknowledge ensures (ACK) that all nodes involved in the message receive everything correctly and in case of error, the transmitter is immediately alerted to send the data packets again. The transmitter sends recessive, the receiver asserts dominant
	CAN 2.0B	1	
ACK Delimiter	CAN 2.0A	1	The single bit must be recessive
	CAN 2.0B	1	
EOF	CAN 2.0A	7	End of Frame (EOF) denotes the end of a current CAN message. Bits must be recessive
	CAN 2.0B	7	

4. Theoretical Background

4.1. Unsupervised SOM Neural Network

The Kohonen Self-Organizing Map (SOM) is a type of Artificial Neural Network (ANN) which allows the visualization of high-dimensional data on a two-dimensional map. The Kohonen SOM is a nonlinear mapping network aimed at computing similarities among data in the input layer and representing them in an output layer of interconnected neurons according to spatial constrains [44].

Most of techniques in a neural network use supervised learning based on back propagation methods for updating weights and error correction learning. Training of supervised networks requires a target variable. The Kohonen SOM network differs from other Artificial Neural Networks since it can be trained by unsupervised learning. The SOM network uses competitive learning in order to find similarities among data, clustering them into different classes of data [45], and it is characterized by a feed-forward structure with a single computational layer [46].

The Kohonen output network consists of a competitive layer where an n-dimensional *codebook* vector is assigned to each neuron in the map and the vector elements represent weights [47]. 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 [48]. Spatial constraints entail that neighboring neurons have similar codebook vectors. Input data vectors are assigned to neurons according to defined measures of distance between them [44] If two different input data vectors are similar, then they will be mapped in neighboring neurons on the network grid. Hence, input data vectors mapped on the same neuron or in the neighboring ones are similar.

The SOM algorithm computes similarities between each input data vector and the neurons' codebook vectors in order to find the most similar. The winning neuron, called *Best Matching Unit* (BMU), adjusts its codebook vectors basing on a weighted average in order to move closer to the input vector. The weight of the attraction between the BMU and the input data vector is one of the training parameters of the model also called *learning rate* α. This parameter changes at each iteration, decreasing during the training process and ensuring the convergence of the model [48]. Additionally, the neighboring neurons adjust their codebook vectors in order to better match with the input vector, thereby ensuring the spatial constraints in order to preserve the topology of the map.

Figure 1 shows a simple illustration of the Kohonen SOM algorithm. After determining the number of neurons in the Kohonen map, a codebook vector is randomly initialized for each neuron. Then, the algorithm computes the distance between a CAN message vector random selected and all the neurons' codebook vectors in the map. The neuron with the smallest distance is the winner, also called the Best Matching Unit. The algorithm also identifies neurons with similar codebook vectors as neighboring neurons. Both the BMU and neighboring neurons are updated in order to move closer to the CAN message vector. The same procedure is repeated for all CAN messages and for a given number of iterations.

The Kohonen SOM network can be trained using the online and the batch algorithms [49]. In the *online* algorithm, the BMU and the neighborhood neurons are adjusted immediately after an input vector is presented to the network. In the *batch* algorithm, the BMU and the neighborhood neurons are updated after all the input data vectors are presented to the network [50].

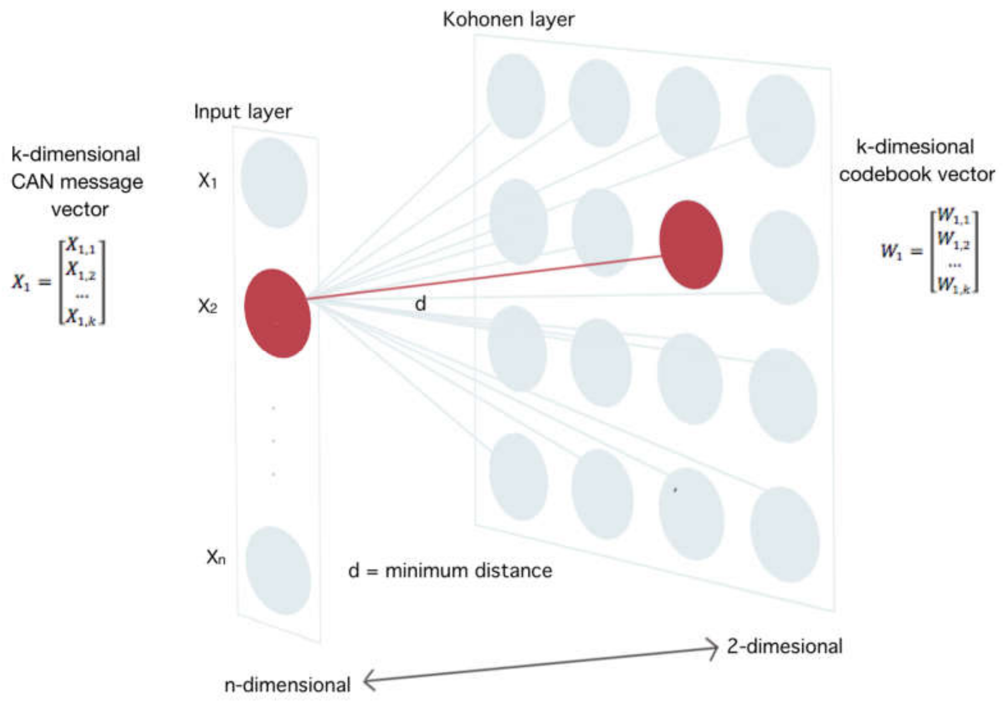
Kohonen layer
Input layer
k-dimensional CAN message vector
$X_1 = \begin{bmatrix} X_{1,1} \\ X_{1,2} \\ \dots \\ X_{1,k} \end{bmatrix}$
X1
X2
Xn
d
k-dimesional codebook vector
$W_1 = \begin{bmatrix} W_{1,1} \\ W_{1,2} \\ \dots \\ W_{1,k} \end{bmatrix}$
d = minimum distance
2-dimesional
n-dimensional

Figure 1. Kohonen Self-Organizing Map (SOM) neural network.

Formally, we defined the input data vectors Xi, with i=1,…,n, the Kohonen neurons Rj, with j=1,…,m, and the codebook vectors Wj with j=1,…,m associated to each neuron. The number of elements of the codebook vector equals the number of variables in the input data vector [51]. We also set the number of iterations t with t=1,…,s. The number of iterations to complete the leaning process is expressed in *epochs*. One epoch indicates the steps of the learning algorithm that allow a complete presentation of the input dataset to the network in order to be learned.

Many different measures of similarity can be used to measure the distance between input data vectors X_i and neurons' codebook vectors W_j [48], such as Manhattan, Tanimoto, Bray Curtis, Canberra, and Chebyshev distances. However, Euclidean distance normally gives slightly better classification results and epoch t can be defined as follows:

$$d_j(t) = \|X_i(t) - W_j(t)\|\,, \tag{1}$$

where Euclidean distance is computed between the CAN message vector Xi, randomly selected, and all the codebook vectors W_j with j=1,…,m on the Kohonen map. Subsequently, the neuron associated to the codebook vector W_j with the minimum distance to X_i is the winning neuron BMU. The distance to the BMU at epoch t is here denoted with the subscript c:

$$d_c(t) = \min_j d_j(t). \tag{2}$$

Once BMU is found, the neuron and its spatial neighbors are updated by the following:

$$W_j(t+1) = W_j(t) + h_{jc}(t)\left[X_i(t) - W_j(t)\right] \tag{3}$$

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

$$h_{jc}(t) = \alpha(t) \cdot \exp\left(\frac{\|r_j - r_c\|^2}{2\sigma(t)^2}\right) \tag{4}$$

where $\alpha(t)$ is the *learning rate function* which is a function monotonically decreasing at each iteration t and r_j is the position of neuron j.

The online training algorithm of the SOM can be implemented using a stepwise recursive procedure. The pseudo-code is 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 W_j associated to neurons R_j with j=1,…,m,

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

```
set($n$, $m$, $s$) // set number of messages, number of neurons, number of epochs
$W_j \longleftarrow \emptyset$
for $j$ = 1,…, $m$ do
    $W_j \longleftarrow$ random($X_i$) // random codebook vectors initialization
end for
set($\alpha$) // set initial learning rate
for $t$ = 1 : $s$ do
    for $i$ = 1 : $n$ do
        $X_i \longleftarrow$ random($X_i$) //random selection of a CAN message data vector
        for $j$ = 1 : $m$ do
            $d_j = \|X_i - W_j\|$ //compute Euclidean distance
        end for
        $d_c = \min_j d_j$ //compute the winning neuron BMU
        for $j$ = 1 : $m$ do
            $W_j = W_j + h_{jc}(\alpha)[X_i - W_j]$ // update neurons
        end for
    end for
    update($\alpha$)
end for
```

The batch algorithm is a variant of the traditional online SOM algorithm. Neurons' codebook vectors are adjusted only after all the input data vectors *Xi* in the input layer are assigned to their winning neuron's BMU in the Kohonen network. Codebook vectors of BMU and neighbors' neurons are updated as follows:

$$W_j(t+1) = \frac{\sum_{i=1}^{n} h_{jc(i)}(t) X_i}{\sum_{i=1}^{n} h_{jc(i)}(t)} \tag{5}$$

where *c(i)* is the index of the winning neuron's BMU for the input data vector X_i and n is the number of input data vectors, at iteration t.

4.2. K-Means Clustering Algorithm

The k-means is one of the most used clustering algorithms for large datasets. It is an unsupervised Machine Learning algorithm which allows one to partition data into *K* groups, minimizing the variance within the clusters. After determining the number of *K* clusters, *K* input data vectors are randomly selected as initial cluster centroids. The Euclidean distance is computed to assign input data vectors to the closest centroids. The *K* centroids are updated and the input data vectors are reassigned at each iteration. These steps are iteratively repeated until input data assignments stop changing and convergence is achieved.

Formally, we defined the input data vectors X_i, with i=1,…,n, the number of clusters *K* and the centroids Φ_k with k=1,…,K. The pseudo-code is detailed in Algorithm 2.

Algorithm 2: K-means algorithm.

```
Input: Input vectors $X_i$ with $i$=1,…, $n$ // Number of desired clusters $K$
Output: Final centroids $\Phi_k$ whit $k$=1, …, $K$
Results: Assign cluster membership to all input vectors $X_i$

$\Phi_k \longleftarrow \emptyset$
set($K$) // set number of clusters
for $k$ = 1,…,$K$ do
    $\Phi_k \longleftarrow$ random($X_i$) // random centroids initialization
end for
repeat
    for $i = 1 : n$ do
        for $k = 1 : K$ do
            $d_k = \|X_i - \Phi_k\|$ // compute Euclidean distances
        end for
        $l = \underset{k}{\operatorname{argmin}}\, d_k$ // compute minimum distance
        $\Phi_l = \Phi_l \cup \{X_i\}$ // assign input vector to the closest cluster centroid
    end for
    $\Phi_k$<- $\Phi_l$ // update centroid $\Phi_k$ based on current partition
until the class assignment converges
return$\{\Phi_1, \dots, \Phi_K\}$
```

5. Materials and Methods

We tested a distance-based intrusion detection system to identify attack and anomaly messages injected on a CAN bus. The intrusion detection system was based on a hybrid unsupervised Kohonen SOM neural network in order to improve the efficiency of the model in detecting attack messages. We proposed a novel distance-based procedure to integrate the unsupervised Kohonen SOM network with the k-means algorithm and compared it with the traditional procedure. Both the methods were tested on open source data containing four datasets, each with a different type of attack: *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 being performed [40,52]. Each dataset contains a total of 30 to 40 minutes of CAN traffic with 300 intrusions of messages injected for 3 to 5 seconds. The Kohonen SOM network was first tested on single datasets separately; then, they were merged into a unique *mixed* dataset containing all types of attack.

In the *DoS* attack database, attack messages with dominant CAN IDs are injected on the CAN bus with the aim of tampering with the accessibility to the network. The *spoofing gear* and spoofing *RPM* datasets contain attack messages concerning, respectively, the driver gear and the RPM gauge aimed at changing the status on the instrument panel. In the *fuzzy* dataset, messages of spoofed random CAN ID and data values are injected in order to damage the vehicle's functionality, due to the manipulation of normal CAN ID and data values.

The experimentation was carried out in the following steps (Figure 2):

1. Data pre-processing: Data were pre-processed in order to be given input to the network. We analyzed open source car hacking data including different kinds of attack messages. In particular, we considered four different datasets containing *DoS* attacks, *spoofing the drive gear*, *spoofing the RPM gauge*, and *fuzzy* attacks. Data are available at [53].
2. Experimental process: We tested the proposed method based on a novel procedure to integrate the Kohonen SOM network with a k-means algorithm in order to improve the performance of the model in terms of accuracy in detection of attack messages and reduction of false negative rate. We also compared the proposed method with the traditional procedure. Both methods were tested, first, on each attack dataset separately, and then on the *mixed* dataset.
3. Evaluation of the performances of the models.

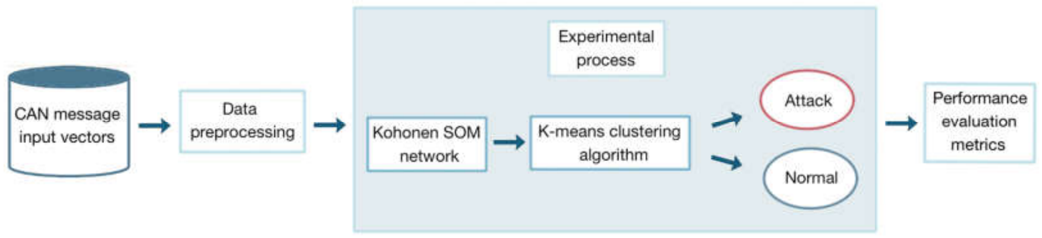
CAN message input vectors
Data preprocessing
Experimental process
Kohonen SOM network
K-means clustering algorithm
Attack
Normal
Performance evaluation metrics

Figure 2. Experimentation process.

5.1. Dataset Pre-Processing

The *DoS* attack, *spoofing the drive gear*, and *spoofing the RPM gauge* datasets present quite regular structures, since messages sent on the CAN bus are characterized by a limited number of different kinds of CAN ID. Figure 3 shows the frequency distribution of the CAN ID identifiers for each dataset and for the *mixed* dataset. The *DoS* dataset presents a very simple structure since it contains normal messages sent with 26 unique CAN IDs and a high frequency of attack messages sent with one different unique CAN ID. *Spoofing the drive gear* and *spoofing the RPM gauge* datasets also show regular structures including a total of 26 unique CAN IDs sending normal messages, one of whom sends attack messages as well.

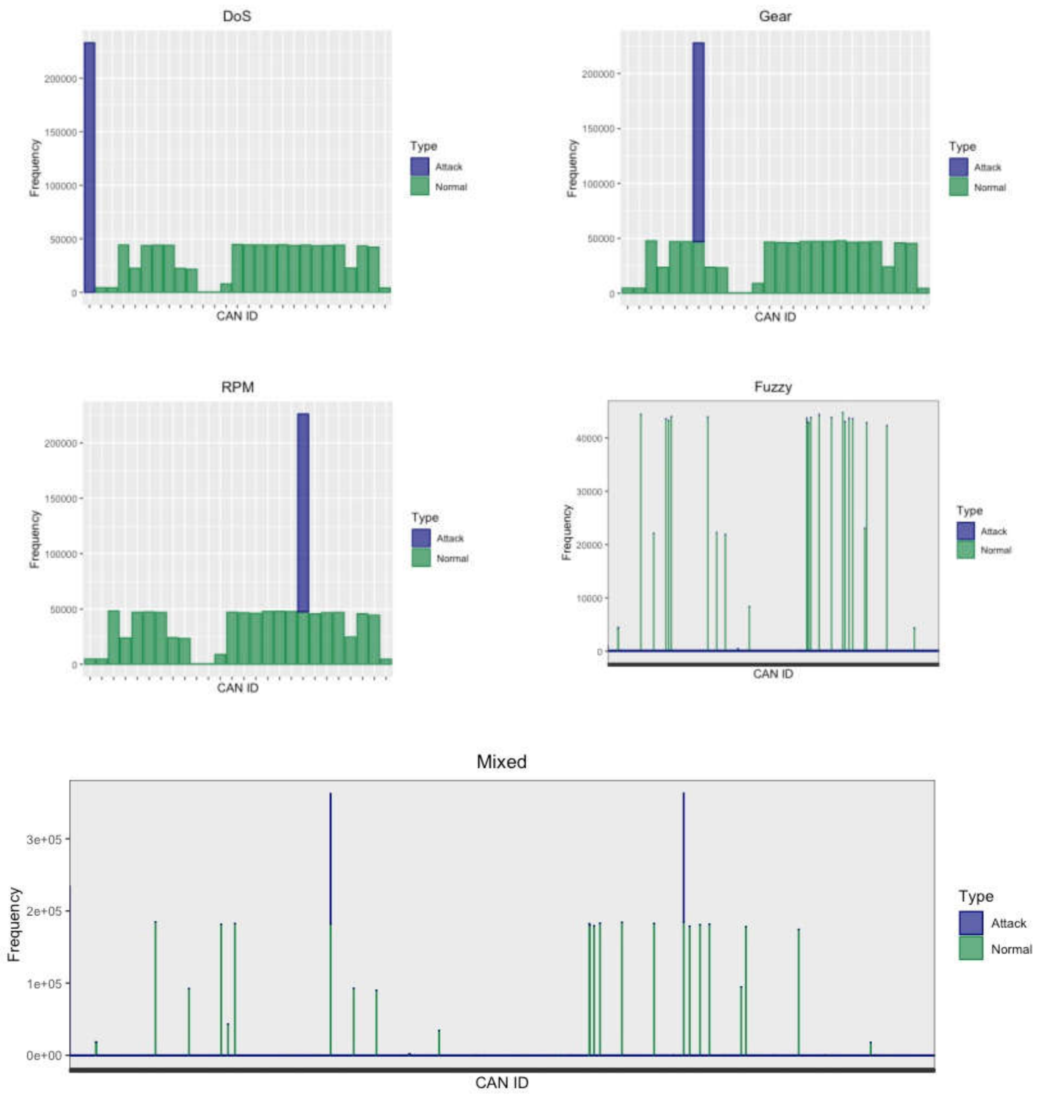

Figure 3. CAN ID frequency distribution.

These three datasets are quite simple to analyze and the proposed model completely succeeded in detecting attack messages when tested on them. The *fuzzy* dataset presents a very complex structure which is really difficult to analyze, since messages are sent on the CAN bus using 2017 different unique CAN IDs. Attack messages are sent using all CAN IDs, just 37 of whom are used to also send normal messages. Moreover, messages sent using 97.1% of unique CAN IDs show a relative frequency of less than 0.1% and are transmitted totally randomly. Hence, the *mixed* dataset, the result of merging all datasets, highlights an even greater complexity in the analysis.

All datasets included the following information: recorded time in seconds, *timestamp*, identifier of CAN message in HEX, *CAN ID*, number of data bytes from 0 to 8, *DLC*, data values, *DATA[0~7]*, and a label R or T which represent, respectively, normal or attack messages (Table 3).

Table 3. 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
….	…	…	…	…	…	…	…	…	…	…	…

In order to run the network, data were properly pre-processed.

CAN ID identifiers and data values were dealt with a semantic approach considering each CAN ID identifier as a category of messages sent by an ECU and data values as the related value information [54]. Since SOM networks can process only numerical data, these categorical data were transformed into a matrix representation using the one-hot encoding technique. Each CAN ID identifier was represented by a column with value 1 if the message was sent with that CAN ID and 0 otherwise. Data values were merged in a unique string representing the value information of the CAN message. The new variable obtained was also transformed into a matrix representation using the one-hot encoding technique.

With regard to time, data were not processed in chronological order, but in relation to CAN messages period. CAN messages can be periodic, sporadic, or aperiodic. Periodic messages occur at regular time intervals, sporadic messages are sent with a minimum time interval, and aperiodic messages are sent at totally random times [41]. Starting from timestamp, we derived a new variable S whose elements Si, with i=1,…,n, where n is the total number of messages, express the time in milliseconds between two successive instances sent on the bus with the same CAN ID identifier. The datasets contained high numbers of periodic, sporadic, and even aperiodic messages with different CAN IDs sent at different times and with different frequencies. In the analysis we included all kinds of messages. The analysis of the *fuzzy* dataset was the most complex since the structure of the dataset was totally random.

Finally, numerical variables, namely, S and DLC, were normalized in order to avoid bias in the training process that can be generated when dealing with very large input vectors [47]. Normalization was obtained using a linear transformation to scale numerical variables to have values between 0 and 1 as follows:

$$Z_n = \frac{Z_0 - Z_{min}}{Z_{max} - Z_{min}}. \quad (6)$$

where Z_0 is the value of the generic numerical variable Z before normalization and Z_n is the new value of Z after normalization. Z_{min} and Z_{max}, respectively, are the minimum and the maximum value of Z in sample data.

We define X_i, with i=1,…,n, input data vectors corresponding to CAN messages sent on the CAN bus. X represents the input layer processed by the SOM network. Labeled data were not included in the analysis since we trained using unsupervised learning (see Table 4).

Table 4. Processed data.

Input X layer									
Input X CAN message vector	**DLC**	**04f0**	**05f0**	**043f**	**…**	**00080067d113**	**14560ff6b000**	**…**	Time
X_1	1	1	0	0	…	0	0	…	0.27
X_2	0.1	0	0	1	…	0	1	…	0.84
X_3	1	0	1	0	…	1	0	…	0.97
…	…	…	…	…	…	…	…	…	…
X_n	0.5	0	0	0	…	0	0		0.01

5.2. System Architecture

After data pre-processing, we tested a distance-based intrusion detection system aimed at identifying attack or anomalous messages injected on the CAN bus. We implemented a hybrid unsupervised Kohonen network in order to classify attack and normal messages based on global and local similarity among input data vectors [44] using two approaches. The models were tested on samples of 10,000 CAN messages vectors for each dataset.

Network initial learning parameters were defined using a trial and error process. Since our goal was to separate input data vectors into two clusters, attack and normal messages, we trained the network on small maps. Map size does not need to be very large, even with a large number of input data vectors. Training large maps is a time-consuming process since all input data vectors are compared with all the neurons in the map [55].

After many trials, on maps of different sizes, we obtained the best performance in terms of prediction accuracy training for the network on a 2 x 2 (four neuron) map using the Gaussian neighborhood function. The learning rate α was initially set to 0.5 linearly decreasing in a training process of 100 epochs. The Euclidean distance was used to compute the similarity between input message data vectors and neurons in the SOM map since it retuned the best results. Data were split between 80.00% for training the model and 20.00% for testing prediction accuracy.

The unsupervised Kohonen network assigned all input data vectors to the four neurons in the Rx map. In order to classify input messages vectors in two clusters, attack and normal, we combined the output of the SOM network with the k-means algorithm using two distinct procedures.

Using the traditional procedure, generally used to cluster local data classified by the Kohonen SOM network [17,22,23], the output of the trained network was given as input to a k-means algorithm (SOMK-C). The algorithm processed the neurons' codebook vectors in order to classify the four neurons into two groups. Input CAN message vectors were then assigned to the same group of the corresponding neuron.

Compared to the traditional approach, we propose a novel procedure to integrate the Kohonen SOM network with the k-means algorithm (SOMK-D) (see Figure 4). The proposed approach relies on the assumption that the Kohonen network classifies input vectors in clusters based on distance between input vectors and neurons. Thus, in the traditional procedure the k-means algorithm processed neurons' codebook vectors, whereas in the proposed approach distance between input vectors and neurons were processed.

The SOMK-D procedure can be illustrated as follows:

- Input CAN message data vectors were weighted depending on their frequency in the whole traffic dataset, since the nature of attack and normal messages sent in the CAN bus differs in terms of structure and frequency.
- Distances between weighted input CAN vectors and neurons in the Rx map were computed training the Kohonen SOM network.
- Distances were used as input of the k-means algorithm.
- The k-means algorithm was implemented to classify input vectors in two clusters based on their distance to the corresponding winning neurons' BMU.

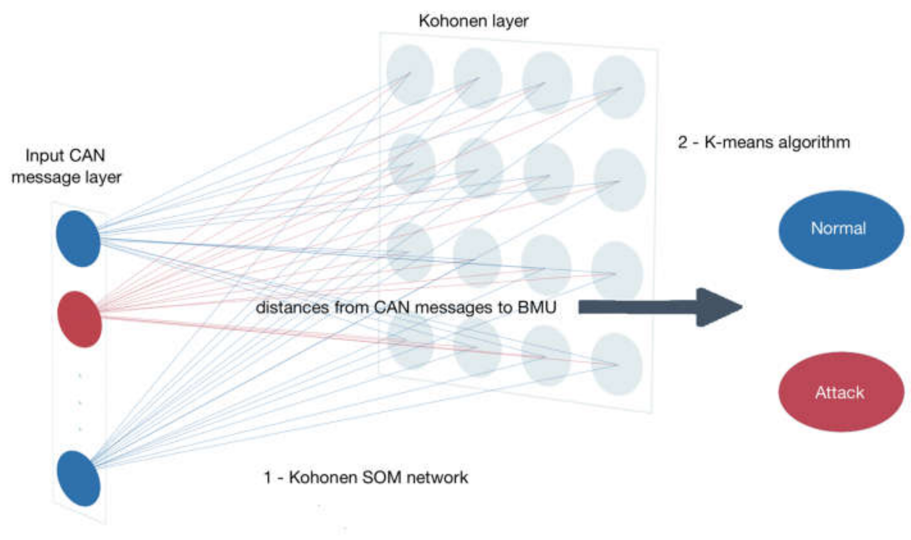

Figure 4. SOMK-D algorithm.

Algorithm 3.1: SOMK-D algorithm.

```
Input: Input layer consisting of weighted CAN message vectors $X_i$ with $i$=1, …,$n$
Output: Output Kohonen layer $R_X$ containing final codebook vectors $W_j$ associated to neurons $R_j$ with $j$ =1,…,$m$
Results: Compute distances between CAN message vectors $X_i$ and winning neurons BMU in the Kohonen map
$W_j \leftarrow \emptyset$
$m \leftarrow 4$ // set number of neurons
for j = 1 : m do
$W_j \leftarrow$ random($X_i$) // random codebook vectors initialization
end for
$\alpha \leftarrow 0.05$ // set initial learning rate
$s \leftarrow 100$ // set initial learning rate
$n \leftarrow 10{,}000$ // set number of CAN messages
for $t = 1 : s$ do
    for $i = 1 : n$ do
        $X_i \leftarrow$ random($X_i$) //random selection of a CAN message vector
        for $j = 1 : m$ do
            $d_j = \|X_i - W_j\|$ //compute Euclidean distance
        end for
        $d_c = \min_j d_j$ //compute the winning neuron $BMU$
        for $j = 1 : m$ do
            $W_j = W_j + h_{jc}[X_i - W_j]$ // update neurons
        end for
    end for
    update($\alpha$)
end for
for $i = 1 : n$ do
    $\delta_i = \|X_i - W_{BMU}\|$ // compute distances from CAN message vectors to winning neurons $BMU$
end for
```

The algorithm is detailed in Algorithms 3 and 4. Results highlight a great improvement in terms of detection accuracy and a significant reduction of the false negative rate compared to the traditional procedure, as shown in Section 6.

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

Algorithm 3.2: SOMK-D algorithm.

```
Input: Distances between CAN message vectors $X_i$ and winning neurons BMU in the Kohonen map // Number of desired clusters $K$
Output: Final centroids $\Phi_k$ whit $k$=1, …, $K$
Results: Assign cluster membership to all input CAN message vectors $X_i$
    $\Phi_k \longleftarrow$ random($X_i$) // random centroids initialization
end for
repeat
    for $i = 1 : n$ do
        for $k = 1 : K$ do
            $d_k = \|\delta_i - \Phi_k\|$ // compute Euclidean distances
        end for
        $l = \underset{k}{\text{argmin}}\, d_k$ // compute minimum distance
        $\Phi_l = \Phi_l \cup \{\delta_i\}$ // assign distances to the closest cluster centroid
    end for
    $\Phi_k \longleftarrow \Phi_l$ // update centroid $\Phi_k$ based on current partition
until the class assignment converges
return{$\Phi_1, \dots, \Phi_K$}
```

5.3. Performance Evaluation Metrics

Traditional classification metrics were used to evaluate the performances of the two explained methods tested on the described datasets.

In particular, given TP (true positive) and TN (true negative)—the numbers of CAN messages correctly classified, respectively, as attack or normal; and FP (false positive) and FN (false negative)—the numbers of CAN messages incorrectly classified, respectively, as attack or normal, we calculated:

$$Accuracy = \frac{TP+TF}{TP+FP+TN+FN}\ , \tag{7}$$

the ratio of correctly classified instances,

$$Precision = \frac{TP}{TP+FP}\ , \tag{8}$$

the ratio of correctly detected errors to the total of detected errors,

$$Recall = \frac{TP}{TP+FN}\ , \tag{9}$$

the ratio of the correctly detected errors to the total of actual errors including not-detected ones,

$$F1 = \frac{2\,(Precision \cdot Recall)}{Precision+Recall}, \tag{10}$$

a weighted average of precision and recall.

We also computed the false negative rate (FNR), that is, the fraction of undetected attacks, as – follows:

$$FNR = \frac{FN}{TP+FN}. \tag{11}$$

The FRN measure has great importance in attack detection for in-vehicle CAN buses and its value should be very small since even a very small number of undetected attacks can cause damage in the vehicle, impairing safety.

6. Results and Discussion

As described in Section 5, we made a distance-based intrusion detection system aimed at identifying attack messages injected into the CAN bus. The system was based on an unsupervised Kohonen SOM network combined with a k-means algorithm using two different approaches. The models were tested first by individually training *DoS*, *spoofing gear*, *spoofing RPM*, and *fuzzy* datasets, and then by merging the four different kinds of attack into a unique *mixed* dataset. The test was conducted on samples of 10,000 CAN messages vectors for each dataset. Samples were randomly selected, balancing the ratio of CAN ID identifiers, and were split into 80.00% training and 20.00% test sets, again stratifying the CAN ID identifiers for all the datasets with the exception of the *fuzzy* dataset. There was a high number of different messages identifiers in the *fuzzy* dataset—2017 unique CAN IDs, 97.1% of whom showed a relative frequency less than 0.1%. Since the nature of attack and normal messages sent in the CAN bus differs in terms of structure and frequency, we weighted input message data vectors depending on their frequency in the whole traffic dataset. The training sets were used to train the algorithms and the test sets to evaluate the models' performances.

We trained the unsupervised Kohonen SOM network on a 2 x 2 (four neuron) map as described in Section 5.2. The network computed the codebook vector for each neuron in the Kohonen map and the input CAN message vectors were assigned to the closest neurons. Figure 5 illustrates results of the unsupervised Kohonen SOM network, respectively, for *DoS*, *spoofing gear,* spoofing *RPM*, *fuzzy,* and *mixed* datasets. In particular, Figure 5 shows input CAN message vectors assigned to neurons in the map by the Kohonen SOM network for test sets. It is evident that attack and normal messages are well separated in *DoS*, *spoofing gear,* and spoofing *RPM* datasets. The distinction is less clear in the *fuzzy* dataset, due to its more complex structure, and, consequently, in the *mixed* dataset. Following the traditional procedure, the codebook vectors computed by the Kohonen SOM network were given as input to the k-means algorithm, in order to cluster the neurons in two groups, attack and normal (SOMK-C). In each dataset, the k-means algorithm clustered three neurons as normal, represented in yellow, and one neuron as attack, the red one. Results for test sets are shown in Figure 6.

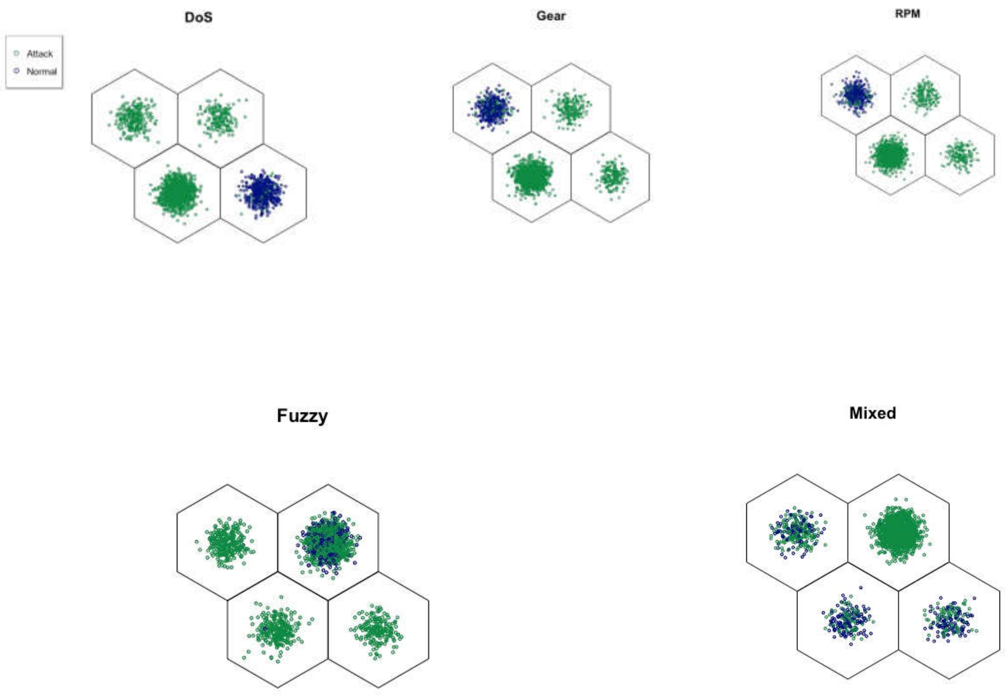

Figure 5. Unsupervised Kohonen SOM network trained on the test set.

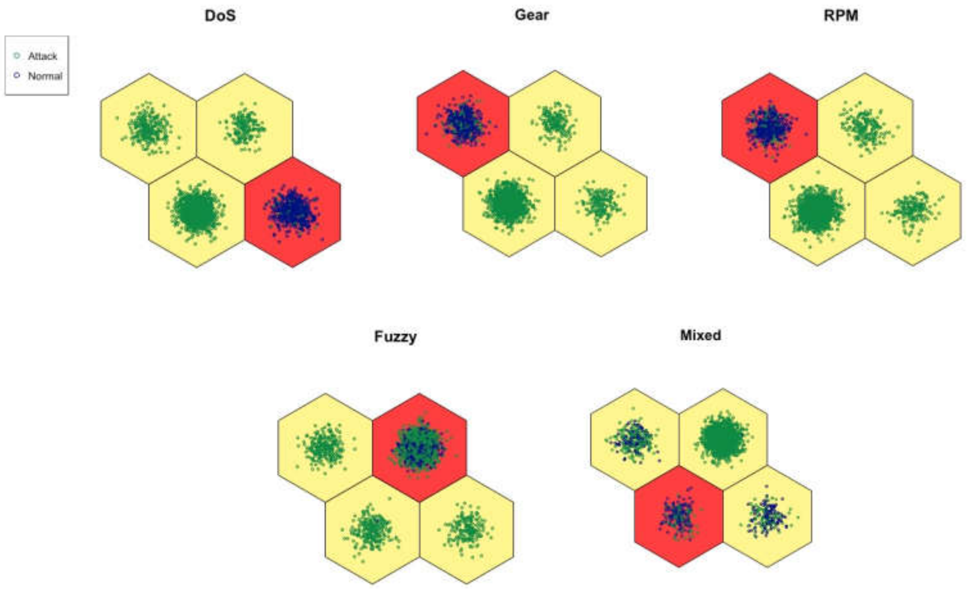

Figure 6. SOMK-C trained on the test set.

Figure 7 represents, for each dataset, the distance between each input CAN message vector and its winning neuron BMU computed by the Kohonen SOM network for the training set. Blue and green points in the plots represent, respectively, attack and normal messages.

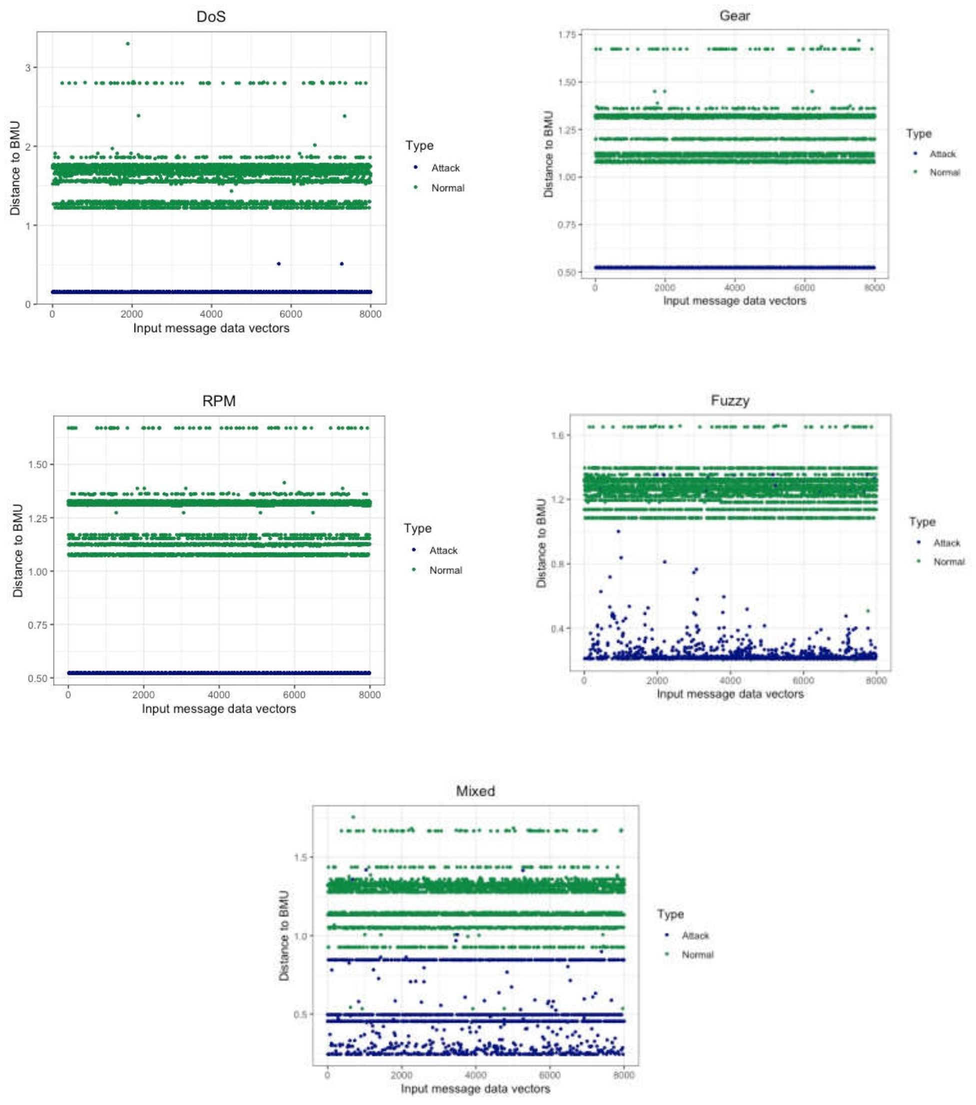

Figure 7. Distances from input CAN message vectors to the winning neurons' Best Matching Unit (BMU) computed by the Kohonen SOM network for the training set.

Table 5 shows distance metrics distinguished for attack and normal messages as classified by the SOMK-C for training set. Plots and related metrics highlight regular patterns and clear separation in distances in *DoS*, *spoofing gear,* and spoofing *RPM* datasets, which were slightly less evident in *fuzzy* and *mixed* datasets. In light of this, we tried to improve the efficiency of the Kohonen SOM network in identifying attack messages using a second procedure that takes into account the evident separation in distances. We tried to enhance the separation between attack and normal messages by clustering distances from input CAN message vectors and corresponding winning BMU neurons. With that aim, after computing Euclidean distances between each input CAN message vector and its winning BMU neuron, we clustered them into two groups using the k-means algorithm (SOMK-D).

Table 5. Distances between input vectors and winning neurons' BMU metrics for attack and normal messages as classified by the SOM network for training set.

Dataset	Type	Min	Median	Mean	Max	Standard deviation
DoS	*Attack*	*0.15*	*0.15*	*0.26*	*2.39*	*0.34*
	Normal	1.22	1.74	1.67	3.30	0.20
Spoofing gear	*Attack*	*0.52*	*0.52*	*0.65*	*1.13*	*0.24*
	Normal	1.08	1.32	1.28	1.72	0.10
Spoofing RPM	*Attack*	*0.52*	*0.52*	*0.65*	*1.13*	*0.25*
	Normal	1.07	1.32	1.28	1.67	0.10
Fuzzy	*Attack*	*0.20*	*1.31*	*1.08*	*1.66*	*0.43*
	Normal	1.09	1.09	1.14	1.26	0.05
Mixed	*Attack*	*0.24*	*1.32*	*1.20*	*1.76*	*0.27*
	Normal	0.94	1.12	1.04	1.12	0.09

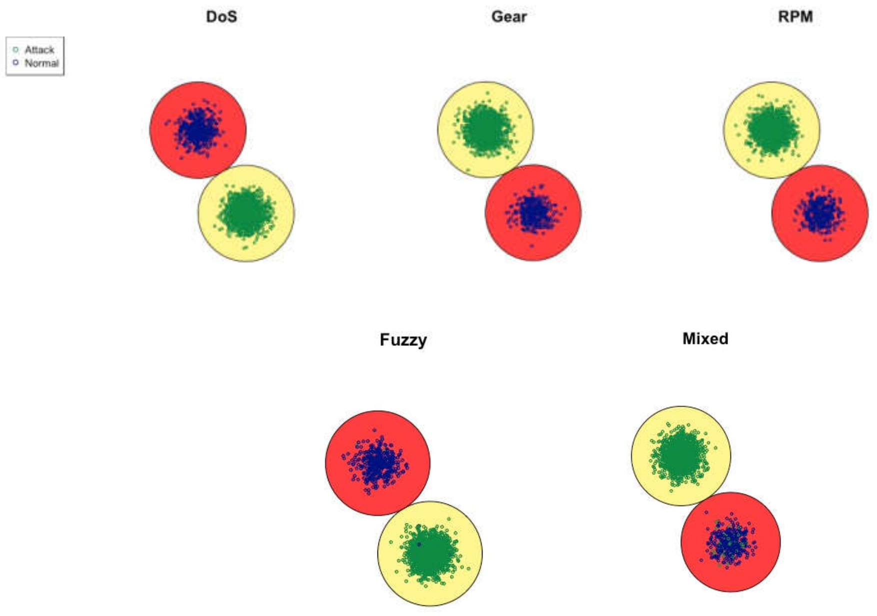

Figure 8. SOMK-D trained on the test set.

Results for the test sets are shown in Figure 8 where it is evident how this procedure greatly improves the efficacy of the model in terms of detecting attack messages—completely succeeding in *DoS*, *spoofing gear,* and spoofing *RPM* datasets.

Evaluation metrics for SOMK-C and SOMK-D procedures are shown in Tables 6 and 7, respectively, for training and test sets. The SOMK-D approach compared to the SOMK-C, improves evaluation metrics results in all the datasets. In SOMK-D, values of accuracy, precision, recall and F1 increase to 100.00% for *DoS, spoofing gear,* and spoofing *RPM* datasets both in training and test sets. The false negative ratio (FNR) is, consequently, equal to 0.00% in the same datasets. In the *fuzzy* dataset, the SOMK-D improved accuracy from 73.20% to 99.58% in the training set and from 72.55% to 99.40% in the test set. Precision and recall increased from 0.46% and 0.20%, respectively, to 99.66% and 98.07% in the training set, and from 0.00% and 0.00%, respectively, to 98.92% and 97.87% in the test set. The false negative ratio (FNR) was reduced from 99.80% to 1.93% in the training set and from 100.00% to 2.13% in the test set. A slight overfitting is noted in the *mixed* dataset.

The Kohonen SOM network was trained for 100 epochs and the output processed by the k-means algorithm converged in less than 10 iterations in all datasets both for SOMK-C and SOMK-D.

Table 6. Evaluation metrics for the training sets.

Dataset		Training set (%)				
		Accuracy	*Precision*	*Recall*	*F1*	*FNR*
DoS	*SOMK-C*	*97.78*	*90.88*	*100.00*	*95.22*	*0.00*
	CI (95%)	*97.45–98.10*	*90.25–91.51*	*100,00–100,00*	*94.75–95.69*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00–100,00	100,00–100,00	100,00–100,00	100,00–100,00	0,00–0,00
Spoofing gear	*SOMK-C*	*95.60*	*79.70*	*100.00*	*88.70*	*0.00*
	CI (95%)	*95.15–96.05*	*78.82–80.58*	*100,00–100,00*	*88.01–89.40*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00-100,00	100,00-100,00	100,00-100,00	100,00-100,00	0,00-0,00
Spoofing RPM	*SOMK-C*	*95.49*	*79.04*	*100.00*	*88.29*	*0.00*
	CI (95%)	*95.03-95.94*	*78.14–79.93*	*100,00–100,00*	*87.59–89.00*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00-100,00	100,00–100,00	100,00–100,00	100,00–100,00	0,00–0,00
Fuzzy	*SOMK-C*	***73.20***	**0.46**	***0.20***	***0.28***	***99.80***
	CI (95%)	*72.23-74.17*	0.32–0.61	*0.10–0.30*	*0.16–0.39*	*99.70–99.80*
	SOMK-D	**99.58**	**99.66**	**98.07**	**98.86**	**1.93**
	CI (95%)	99.43-99.72	99.54–99.79	97.77–98.37	98.63–99.09	1.63–2.23
Mixed	*SOMK-C*	*82.55*	*61.76*	*27.91*	*38.45*	*72.09*
	CI (95%)	*81.72-83.38*	*60.69–62.82*	*26.93–28.90*	*37.38–39.51*	*71.10–73.07*
	SOMK-D	99.86	99.68	99.62	99.65	0.38
	CI (95%)	99.78-99.94	99.56–99.80	99.48–99.75	99.52–99.78	0.25–0.52

Table 7. Evaluation metrics for test sets

Dataset		Test set (%)				
		Accuracy	*Precision*	*Recall*	*F1*	*FNR*
DoS	*SOMK-C*	*97.85*	*91.21*	*100.00*	*95.40*	*0.00*
	CI (95%)	*97.22–98.49*	*89.97–92.45*	*100,00–100,00*	*94.48–96.32*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00-100,00	100,00–100,00	100,00–100,00	100,00–100,00	0,00–0,00
Spoofing gear	*SOMK-C*	*95.50*	*79.36*	*100.00*	*88.49*	*0.00*
	CI (95%)	*94.59–96.41*	*77.58–81.13*	*100,00–100,00*	*87.09–89.89*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00-100,00	100,00–100,00	100,00–100,00	100,00–100,00	0,00–0,00
Spoofing RPM	*SOMK-C*	*95.75*	*80.41*	*100.00*	*89.14*	*0.00*
	CI (95%)	*94.87–96.63*	*78.68–82.15*	*100,00–100,00*	*87.78–90.51*	*0,00–0,00*
	SOMK-D	100.00	100.00	100.00	100.00	0.00
	CI (95%)	100,00-100,00	100,00–100,00	100,00–100,00	100,00–100,00	0,00–0,00
Fuzzy	*SOMK-C*	***72.55***	***0.00***	***0.00***	***-***	***100.00***
	CI (95%)	*70.59–74.51*	*0,00–0,00*	*0,00–0,00*	*–*	*100,00–100,00*
	SOMK-D	**99.40**	**98.92**	**97.87**	**98.39**	**2.13**

	CI (95%)	99.06–99.74	98.47–99.37	97.23–98.50	97.84–98.94	1.50–2.77
Mixed	*SOMK-C*	*87.01*	*61.05*	*38.41*	*47.15*	*61.59*
	CI (95%)	*85.53-88.48*	*58.92–63.19*	*36.28–40.54*	*44.97–49.34*	*59.46–63.72*
	SOMK-D	97.60	86.29	100.00	92.64	0.00
	CI (95%)	96.93-98.27	84.78–87.79	100,00–100,00	91.49–93.78	0,00–0,00

7. Conclusions

The great diffusion of embedded and portable communication devices on modern vehicles and smart transport systems enable communication with internal and external devices, networks, applications, and services. As vehicle connectivity becomes common, new security risks emerge, since communication protocols, such as CAN network, are still insecure and vulnerable to attacks. For this reason, there is an increasing interest in automotive cybersecurity for in-vehicle communication systems [56].

In this work we propose a distance-based intrusion detection system based on an unsupervised Kohonen SOM network. The SOM network found general applications in security issues because of its features of high detection rate, short training time, and high versatility.

In our work, we introduced the Kohonen SOM network as an intrusion detection system for in-vehicle communication networks aimed at identifying attack messages injected on a CAN bus. In a previous paper we showed the results of the implementation of a hybrid supervised Kohonen SOM network combined with other techniques for clustering in order to improve the performance of the network. In the present work we implemented an unsupervised Kohonen SOM network combined with a k-means algorithm in order to improve the efficiency of the model in detecting attack messages. We proposed a novel distance-based procedure to integrate the unsupervised Kohonen SOM network with the k-means algorithm and compared the proposed method to the traditional procedure.

We tested both the methods on open source data containing four different datasets: *DoS* attack, *spoofing the drive gear*, *spoofing the RPM gauge*, and *fuzzy* attack. We first used the networks on separate datasets with single kinds of attack, and then merged them all together in a *mixed* dataset. The *mixed* dataset presented a highly complex structure to analyze, characterized by large volume of traffic with low number of features and highly imbalanced data distribution with more than 2000 different attack types sent totally randomly. Despite the complex structure of the CAN network dataset, the proposed method showed high performance in detection accuracy—completely succeeding in *DoS* attack, *spoofing the drive gear*, and *spoofing the RPM* gauge datasets. Moreover, it significantly reduced the false negative rate, which has great importance in attack detection for in-vehicle CAN buses in order to ensure the safety of the vehicle.

Due to the availability of the open source data, this work only tested the proposed IDS on a limited number of attacks. Future works should involve investigating the development of the Kohonen SOM network as an anomaly detector by extending it to other types of attacks in a CAN bus.

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. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Vasudev, H.; Das, D.; Vasilakos, A.V. Secure message propagation protocols for IoVs communication components. *Comput. Electr. Eng.* **2020**, *82*, 106555, doi:10.1016/j.compeleceng.2020.106555.
2. Du, R.; Santi, P.; Xiao, M.; Vasilakos, A.V.; Fischione, C. The Sensable City: A Survey on the Deployment and Management for Smart City Monitoring. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 1533–1560, doi:10.1109/comst.2018.2881008.
3. Barletta, V.; Caivano, D.; DiMauro, G.; Nannavecchia, A.; Scalera, M. Managing a Smart City Integrated Model through Smart Program Management. *Appl. Sci.* **2020**, *10*, 714, 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 AEIT International Annual Conference; Institute of Electrical and Electronics Engineers (IEEE), 2018; pp. 1–6,.
5. Zhou, J.; Dong, X.; Cao, Z.; Vasilakos, A.V. Secure and Privacy Preserving Protocol for Cloud-Based Vehicular DTNs. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1299–1314, doi:10.1109/tifs.2015.2407326.
6. Baldassarre, M.T.; Barletta, V.; Caivano, D.; Scalera, M. Integrating security and privacy in software development. *Softw. Qual. J.* **2020**, 1–32, doi:10.1007/s11219-020-09501-6.
7. Zhou, J.; Cao, Z.; Dong, X.; Vasilakos, A.V. Security and Privacy for Cloud-Based IoT: Challenges. *IEEE Commun. Mag.* **2017**, *55*, 26–33, doi:10.1109/mcom.2017.1600363cm.
8. Challa, S.; Das, A.K.; Gope, P.; Kumar, N.; Wu, F.; Vasilakos, A.V. Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems. *Futur. Gener. Comput. Syst.* **2020**, *108*, 1267–1286, doi:10.1016/j.future.2018.04.019.
9. Sommer, F.; Duerrwang, J.; Kriesten, R. Survey and Classification of Automotive Security Attacks. *Inf.* **2019**, *10*, 148, doi:10.3390/info10040148.
10. Caivano, D. Continuous Software Process Improvement through Statistical Process Control. In Proceedings of the Ninth European Conference on Software Maintenance and Reengineering; Institute of Electrical and Electronics Engineers (IEEE), 2005; pp. 288–293.
11. Baldassarre, M.T.; Boffoli, N.; Caivano, D.; Visaggio, G. Managing Software Process Improvement (SPI) through Statistical Process Control (SPC). *Intelligent Tutoring Systems* 2004, *3009*, 30–46.
12. Baldassarre, M. T.; Barletta, V. S.; Caivano, D.; Raguseo, D.; Scalera, M.; Teaching cyber security: The hack-space integrated model, CEUR Workshop Proceedings, In ITASEC, 2019, 2315.
13. 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.
14. Carsten, P.; Andel, T.R.; Yampolskiy, M.; McDonald, J.T. In-Vehicle Networks. In Proceedings of the Proceedings of the 10th Annual Cyber and Information Security Research Conference on - CISR '15; Association for Computing Machinery (ACM), 2015; Vol. 06-08-Apri, pp. 1–8.
15. Gmiden, M.; Gmiden, M.H.; Trabelsi, H. An intrusion detection method for securing in-vehicle CAN bus. In Proceedings of the 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA); Institute of Electrical and Electronics Engineers (IEEE), 2016; pp. 176–180.
16. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of Automotive Controller Area Network Intrusion Detection Systems. *IEEE Des. Test* **2019**, *36*, 48–55, doi:10.1109/mdat.2019.2899062.
17. Qu, X.; Yang, L.; Guo, K.; Ma, L.; Sun, M.; Ke, M.; Li, M. A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection. *Mob. Networks Appl.* **2019**, 1–22, doi:10.1007/s11036-019-01353-0.
18. 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, October 2018; Vol. 435, p. 12009.
19. Wu, Y.; Yan, P.F. A study on structural adapting self-organizing neural network. *Acta Electronica Sinica*, **1999**, *27*, 56–59.
20. Wan, Q.; Wang, C.; Feng, Z. Y.; Ye, J.F. Review of K-means clustering algorithm. *Electronic Design Eng.* **2012**, *20*, 21–24.
21. Feyereisl, J.; Aickelin, U. Self-Organising Maps in Computer Security. *arXiv preprint arXiv:1608.01668*, **2016.**

22. Ling, T.; Chong, L.; Jingming, X.; Jun, C. 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.
23. Huai-Bin, W.; Hong-Liang, Y.; Zhi-Jian, X.; Zheng, Y. A Clustering Algorithm Use SOM and K-Means in Intrusion Detection. In Proceedings of the 2010 International Conference on E-Business and E-Government; Institute of Electrical and Electronics Engineers (IEEE), 2010; pp. 1281–1284.
24. Dhanabal, L.; Shantharajah, S.P. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. *Int. J. Adv. Res. Comput. Commun. Eng*. **2015**, *4*, doi 10.17148/IJARCCE.2015.4696.
25. De La Hoz, E.; De La Hoz, E.; Ortiz, A.; Ortega, J.; Martínez-Álvarez, A. Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. *Knowledge-Based Syst.* **2014**, *71*, 322–338, doi:10.1016/j.knosys.2014.08.013.
26. Palomo, E.J.; Domínguez, E.; Luque, R.M.; Muñoz, J.; Luque-Baena, R.M. Network Security Using Growing Hierarchical Self-Organizing Maps. *Intelligent Tutoring Systems* 2009, *5495*, 130–139.
27. Ippoliti, D.; Zhou, X. A-GHSOM: An adaptive growing hierarchical self organizing map for network anomaly detection. *J. Parallel Distrib. Comput.* **2012**, *72*, 1576–1590, doi:10.1016/j.jpdc.2012.09.004.
28. Zhang, Y.; Bu, W.; Su, C.; Wang, L.; Xu, H. Intrusion detection method based on improved growing hierarchical self-organizing map. *Trans. Tianjin Univ.* **2016**, *22*, 334–338, doi:10.1007/s12209-016-2737-4.
29. 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.
30. Liang, L.; Ye, H.; Li, G.Y. Toward Intelligent Vehicular Networks: A Machine Learning Framework. *IEEE Internet Things J.* **2019**, *6*, 124–135, doi:10.1109/jiot.2018.2872122.
31. Ye, H.; Liang, L.; Li, G.Y.; Kim, J.; Lu, L.; Wu, M.; Li, Y. Machine Learning for Vehicular Networks: Recent Advances and Application Examples. *IEEE Veh. Technol. Mag.* **2018**, *13*, 94–101, doi:10.1109/mvt.2018.2811185.
32. Jing, Q.; Vasilakos, A.V.; Wan, J.; Lu, J.; Qiu, D. Security of the Internet of Things: perspectives and challenges. *Wirel. Networks* **2014**, *20*, 2481–2501, doi:10.1007/s11276-014-0761-7.
33. Baldassarre, M.T.; Barletta, V.; Caivano, D.; Scalera, M. *Privacy Oriented Software Development*; Springer Science and Business Media LLC, 2019; pp. 18–32;.
34. Bezemskij, A.; Loukas, G.; Gan, D.; Anthony, R.J. Detecting Cyber-Physical Threats in an Autonomous Robotic Vehicle Using Bayesian Networks. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData); Institute of Electrical and Electronics Engineers (IEEE), 2017; pp. 98–103.
35. Kang, M.-J.; Kang, J.-W. A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security. In Proceedings of the 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring); Institute of Electrical and Electronics Engineers (IEEE), 2016; Vol. 2016-July, pp. 1–5.
36. Taylor, A.; Leblanc, S.P.; Japkowicz, N. Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA); Institute of Electrical and Electronics Engineers (IEEE), 2016; pp. 130–139.
37. Kalash, M.; Rochan, M.; Mohammed, N.; Bruce, N.D.B.; Wang, Y.; Iqbal, F. Malware Classification with Deep Convolutional Neural Networks. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS); Institute of Electrical and Electronics Engineers (IEEE), 2018; pp. 1–5.
38. Lin, Z.; Shi, Y.; Xue, Z. IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection, *arXiv preprint arXiv* **2018**, *1809*, 02077. https://arxiv.org/abs/1809.02077.
39. Torres, J.M.; Comesaña, C.I.; 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.
40. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198, doi:10.1016/j.vehcom.2019.100198.
41. 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.

42. Cook, J. A.; Freudenberg, J. S. Controller Area Network (CAN). *EECS461* **2008**, 1–8.
43. Chen, S.-H.; Lin, C.-H.R. Evaluation of DoS Attacks on Vehicle CAN Bus System. In Proceedings of the Human Centred Intelligent Systems; Springer Science and Business Media LLC, 2018; pp. 308–314.
44. Barbieri, N. Fuel prices and the invention crowding out effect: Releasing the automotive industry from its dependence on fossil fuel. *Technol. Forecast. Soc. Chang.* **2016**, *111*, 222–234, doi:10.1016/j.techfore.2016.07.002.
45. Ciaburro, G.; Venkateswaran, B. *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*; Packt Publishing Ltd – Livery Place – 35 Livery Street – Birmingham – B3 2PB - UK, 2017; ISBN 978-1-78839-787-2.
46. Akinduko, A.A.; Mirkes, E.M. Initialization of Self-Organizing Maps: Principal Components Versus Random Initialization. A Case Study. *A case study* **2012**, 3–20.
47. Shamsuddin, S. M.; Zainal, A.; Mohd Yusof, N. Multilevel Kohonen Network Learning For Clustering Problems, *J. Inf. Commun. Technol.* **2008**, *7*, 1–25.
48. Wehrens, R.; Buydens, L.M. Self- and Super-organizing Maps in R : The kohonen Package. *J. Stat. Softw.* **2007**, *21*, 1–19,, doi:10.18637/jss.v021.i05.
49. Kohonen, T.; Self-Organizing Maps, Berlin, Heidelberg: Springer Berlin Heidelberg, Germany, 2001; Vol. 30.
50. Wehrens, R.; Kruisselbrink, J. Flexible Self-Organizing Maps in kohonen 3.0. *J. Stat. Softw.* **2018**, *87*, 1–18, doi:10.18637/jss.v087.i07.
51. 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.
52. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST); Institute of Electrical and Electronics Engineers (IEEE), 2018; pp. 1–6.
53. Car-Hacking Dataset - Hacking and Countermeasure Research Lab." Available online: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset. (accessed: 27 November 2019).
54. Zhou, A.; Li, Z.; Shen, Y.; Zhou; Li; Shen Anomaly Detection of CAN Bus Messages Using A Deep Neural Network for Autonomous Vehicles. *Appl. Sci.* **2019**, *9*, 3174, doi:10.3390/app9153174.
55. Nakayama, K.; Matsuo, Y. MIGSOM: A SOM Algorithm for Large Scale Hyperlinked Documents Inspired by Neuronal Migration. *Intelligent Tutoring Systems* 2014, *8421*, 79–94.
56. Han, K.; Weimerskirch, A.; Shin, K. Automotive Cybersecurity for In-Vehicle Communication, *IQT Q.* **2014**, *6*, 22–25.