

Article

Unsupervised Anomaly Detection for Cars CAN Sensors Time Series Using Small Recurrent and Convolutional Neural Networks

Yann Cherdo ^{1,2}, Benoit Miramond ², Alain Pegatoquet ^{2,*}  and Alain Vallauri ¹

¹ Renault Software Labs, 2600 Route des Crêtes, Sophia Antipolis, 06560 Valbonne, France; yann.cherdo@univ-cotedazur.fr (Y.C.)

² LEAT (CNRS), Bât. Forum, Campus SophiaTech 930 Route des Colles, 06903 Sophia Antipolis, France

* Correspondence: alain.pegatoquet@univ-cotedazur.fr

Abstract: Predictive maintenance in the car industry is an active field of research for machine learning and anomaly detection. The capability of cars to produce time series data from sensors is growing as the car industry is heading towards more connected and electric vehicles. Unsupervised anomaly detectors are therefore very adapted to process those complex multidimensional time series and highlight abnormal behaviors. We propose to use recurrent and convolutional neural networks based on unsupervised anomaly detectors with simple architectures on real, multidimensional time series generated by the car sensors and extracted from the Controller Area Network bus (CAN). Our method is then evaluated through known specific anomalies. As the computational costs of Machine Learning algorithms are a rising issue regarding embedded scenarios such as car anomaly detection, we also focus on creating anomaly detectors that are as small as possible. Using a state-of-the-art methodology incorporating a time series predictor and a prediction-error-based anomaly detector, we show that we can obtain roughly the same anomaly detection performance with smaller predictors, reducing parameters and calculations by up to 23% and 60%, respectively. Finally, we introduce a method to correlate variables with specific anomalies by using anomaly detector results and labels.



Citation: Cherdo, Y.; Miramond, B.; Pegatoquet, A.; Vallauri, A. Unsupervised Anomaly Detection for Cars CAN Sensors Time Series Using Small Recurrent and Convolutional Neural Networks. *Sensors* **2023**, *23*, 5013. <https://doi.org/10.3390/s23115013>

Academic Editors: Jinoh Kim and Alexander Sim

Received: 23 March 2023

Revised: 28 April 2023

Accepted: 15 May 2023

Published: 23 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: anomaly detection; sensors; Internet of Things; unsupervised; Controller Area Network bus; car; time series; recurrent neural network; long short-term memory; gated recurrent unit; convolutional neural network; computational costs; anomaly likelihood

1. Introduction and Related Work

Predictive maintenance in the car industry refers to the anticipation and management of failures and abnormalities occurring in a car. For example, if the oil pressure remains too low for a long period of time, while it will not cause any immediate damage, it might induce motor and mechanical parts' failures and breaks in the near future. The user will then have to go to the garage, broken parts will be replaced, and a root cause of the failure will be identified, if possible. Anticipating and understanding such a failure occurrence can bring about the following outcomes. First, comprehension of the root causes of part failures allows for better design and manufacturing, enhancing the generation quality of future products. Second, anticipating a part failure allows for better purchase management and logistics for the replacement parts, resulting in financial optimization. Finally, it helps the garage to better solve the issue by providing more information and knowledge about the failure.

One can comprehend a vehicle as a complex electric and mechanical system that has an inner control and sensing system. In Section 2, we offer more detailed information about car architecture. The data produced by sensors, for example the oil pressure or the battery voltage, can be extracted and then processed off-board. This results in multidimensional time series data. As the car industry progresses towards more connected, electrical, and

complex vehicles, the amount of data produced by sensors that can be processed for predictive maintenance grows both in quality and quantity.

While ML algorithms are being used to process such information on the cloud, two main costs are being generated: the infrastructure cost due to algorithm computations and the bandwidth cost due to data transfers from cars to the cloud. Bandwidth costs and data storage capacities can be blocking constraints in certain cases. Personal data protection can also be problematic when trying to store personal cars' usage data in the cloud. For all of these reasons, getting the ML algorithms closer to the data producer, namely the car, as an embedded system is a rising research topic. The main constraint is the low computational capacity that is generally available in a vehicle. In this paper, we address this issue in the context of unsupervised anomaly detection by lowering the size of the ML algorithm while maintaining the same detection performance. The algorithm itself being a source of complexity, we chose the simplest neural networks from among the best time series anomaly detectors in the literature.

Building supervised machine learning algorithms to predict failures from sensors' time series would be ideal. However, obtaining the labels of failures is not so straightforward and, in certain cases, can introduce significant noise. In fact, failures are not directly stored as a statistic, but instead, one can record parts that have been replaced at the garage and try to predict parts which will need to be replaced. When a vehicle is handled in a garage because of a failure, replaced parts are not always linked to the real failure due to expertise error and habits. Therefore, only a subset of possible failures can be covered by this kind of approach. In practice, datasets of parts' replacements tended to be heavily unbalanced.

On the other hand, the production of failures labels by experts would be very expensive considering the complexity of parts and car models' and scenarios' variations. Unsupervised anomaly detection could therefore be considered as a solution to this problem.

Anomaly Detection is a wide research field with many different techniques [1,2]. Unsupervised anomaly detection refers to the problem of finding abnormal patterns in the data. In order to apply unsupervised anomaly detection within time series, companies such as Amazon, Twitter, Etsy, and Yahoo have developed their own models that are a mixture of classical statistical, decomposition, and machine learning algorithms [3–6]. Some models are also bio-inspired, e.g., modeling the episodic memory of the cortex [7] using a model called hierarchical temporal memory (HTM) [8]. Another bio-inspired model has been presented in Reference [9]. In this work, an evolving spiking neural network (OeSNN) is trained similarly to the HTM to learn sequences continuously. Both HTM and OeSNN detect two kinds of anomalies. The first one is detected when the model predicts the wrong value, and the second one is detected when the model is unable to predict anything from the given context. Those models use local learning rules like Hebbian or genetic rules that have been shown to be less powerful than the well-established gradient descent optimization. In particular, these methods suffer from critical hyper parameters' selection.

Convolutional neural networks (CNN) have been used to efficiently learn normal patterns and reveal anomalous ones [10]. This model was also successfully applied to spectral residual in Reference [11]. The field of graph theory also proposed a graph- and attention-based anomaly detector for multivariate time series in Reference [12].

Heavier models like generative adversarial networks (GAN) based on bidirectional Long Short Term Memory (LSTM) were proposed in Reference [13]. Modeling such correlations over time, or building a temporal model of time series, is an active area of development with manifold solutions [14]. One of the state-of-the-art approaches is LSTM [15]-based anomaly detection, which has had a great impact on the field of unsupervised anomaly detection [16]. In Reference [17], LSTM neural networks were used to predict time series after training on normal time series without any anomalies. A threshold was then applied to the error between the learned prediction and the true time series in order to find anomalous patterns. Some simple one-layer LSTM networks can be used as in Reference [16], although stacked (or deep) LSTM networks with several layers show better results, as demonstrated in References [17–21]. Gated recurrent unit (GRU) [22] is a lighter version of

LSTM also in the family of recurrent neural networks (RNN), and it shows very similar performance in time series modeling. We propose to compare LSTM and GRU models in terms of detection performance and computational costs in order to further our goal of lowering the computational cost of our anomaly detector.

In the car industry, HTM have been used to conduct unsupervised anomaly detection on time series using nine variables provided by racing cars' recordings [8]. Several papers can be found about intrusion detection, safety, and cybersecurity, as connected cars are exposed to hacks [23–25]. Intrusion detection conducted using a complex CNN-LSTM and attention mechanisms in CAN traffic logs can be found in Reference [26]. Within the same application, a multilayer perceptron was employed in Reference [27]. These works focus on the way information transits through the CAN and not on the information itself, as this is sufficient to detect an intrusion. Fewer works have been found that focus directly on the information provided by the car sensors and the commands that transit through the CAN, such as oil pressure, engine temperature, etc. This approach can be used for intrusion detection but also allows for the detection of abnormal complex behavior happening inside the car. In Reference [28], a more classical and statistical method, the Kalman filter, is employed to model such time series and to detect any that are abnormal. In the same context, hidden Markov models have been used to find abnormal internal behaviors [29]. For this kind of car sensor data, we propose the use of LSTM-, GRU- and CNN-based anomaly detectors.

One can find a large community of researchers working on image and video classification for autonomous driving cars. This field also includes anomaly detection, as is exhaustively presented in Reference [30]. Finally, an original approach that uses acoustic recordings inside the car to find anomalies is proposed using feature extractions and deep autoencoders in Reference [31]. It can be seen that most of the state-of-the-art unsupervised anomaly detection algorithms have yet not been applied to car sensors' time series. The goal of this paper is to prove that such a task can be achieved using small yet powerful anomaly detectors. This is why, among all available models, LSTM, GRU, and CNN were chosen for use. These neural networks allow architectures to be kept simple while using the powerful gradient descent optimization method.

In this paper, our contribution is to apply state-of-the-art LSTM- and CNN-based, unsupervised anomaly detection to real car time series and to evaluate the results with respect to abnormal labels. We do not focus on any specific kind of anomaly, and our solution aims at highlighting any complex multidimensional behavior of a vehicle. We then compare our models with different amounts of layers and cells and show that similar anomaly detection performance can be obtained using smaller models even if this limits their prediction accuracy. We finally introduce a method of correlating variables to those specific anomalies using the anomaly detector results and labels.

Our paper is organized as follows. In Section 2, we introduce the data that we are working on, focusing on how they have been produced and extracted from the vehicle and the workflow leading to anomaly detection. In Section 3, we describe in detail the addressed issue and the three state-of-the-art models and architectures we used, including the entire process of unsupervised anomaly detection. In Section 4, we specify the performance metrics we used to evaluate our models and the qualitative results we obtained from the aforementioned data. In this section, we also introduce and evaluate a method for finding variables that are the most correlated with the abnormal labels. In the last section, we discuss the overall work presented in this paper in terms of opportunities and limitations, focusing on our main contributions. We then introduce possible future work that could enhance our models based on our findings.

2. Car Time Series Extracted from the CAN Bus

In this section, we present the process of data generation in the car and how it can be exploited for machine learning purposes.

2.1. The Data and the Car

A car is a complex assembly of mechanical and electrical parts. The control and sensing of all of those parts are processed by electric computing units (ECU). ECUs are microcontroller-based units that can receive, process, and send information from and to other ECUs. Specifically, information transits through a controller area network (CAN) bus. For example, one ECU might control the oil temperature while receiving and processing information about the motor rotational speed that has been sent from the ECU managing the motor area. In a modern car, tens of ECUs can process information simultaneously while the car is running. Information transiting through the CAN bus is thus a very good reflection of the car's global state. In order to extract this information, a spy system can listen to the information transiting on the CAN bus and save it in a structured manner, associating a sample to its variable and timestamp. Then, the resulting data can be sent to a cloud database and into an anomaly detector, as shown in Figure 1.

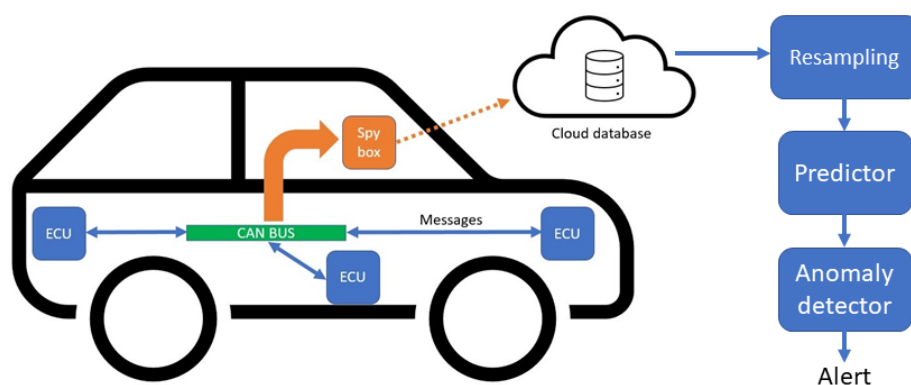


Figure 1. Flow chart presenting the process of data generation in a modern car leading to anomaly detection. Within the car, ECUs handle sensors and communicate some of their values through the CAN bus. That CAN bus can be read in order to extract those sensors' data and can then be sent to the cloud or be exploited internally by another ECU. The resulting time series are then resampled, a prediction is emitted after each sample, and an anomaly is either detected or not given the prediction error.

2.2. Our Dataset

The dataset used in this paper consists of the recordings of 486 variables gathered from an Alpine Renault car during driving tests conducted on circuits. Those tests were conducted over the course of 4 months, resulting in 17 GB of data. The sampling rate of the recordings was 10 Hz. In order to simplify the training and evaluation of the model, we filtered variables with respect to their complexity and retained only the richest ones. As many variables correspond to commands with only one to three discrete values and are rarely triggered, most of the signals show very few variations through time and are therefore poorly informative. Those are the variables which we filtered out. This process resulted in an 85-variable system. As recordings do not have the same temporal length, this dataset can be seen as a set of multivariate time series with different lengths. Finally, we resampled the data to 1 Hz using the average in order to save time on training and testing. This dataset is not public but is necessary in order to apply state-of-the-art anomaly detectors to real case studies. Nevertheless, we have included a comparison of different state-of-the-art models as well as the ones we have chosen in Table 1. These were applied to an open-source benchmark in order to ensure that our method and implementation were valid.

Table 1. Comparison of state-of-the-art anomaly detectors on the public benchmark Yahoo [32]. The LSTM (ours) corresponds to the LSTM with 50 cells and two layers, as presented in Section 4, using the semi-supervised threshold. A1 to A4 correspond to different groups of time-series that can be found within the Yahoo benchmark.

Model	A1	A2	A3	A4	Mean
Hierarchical Temporal Memory (HTM) [33]	0.59	0.66	0.33	0.29	0.47
Online evolving Spiking Neural Network (OeSNN-UAD) [9]	0.70	0.69	0.41	0.34	0.54
Deep learning-based Anomaly detection approach for Time-series (DeepAnT) using a Long Short Term Memory (LSTM) [10]	0.44	0.97	0.72	0.59	0.68
Deep learning-based Anomaly detection approach for Time-series (DeepAnT) using a Convolutional Neural Network (CNN) [10]	0.46	0.94	0.87	0.68	0.74
Time-series Anomaly Detection using Generative Adversarial Networks (TadGAN) [13]	0.8	0.87	0.69	0.6	0.74
Long Short Term Memory (LSTM) (ours)	0.71	1	0.82	0.76	0.82

2.3. Labels

The utilized labels consist of 3683 timestamps in which an anomaly linked to the oil pressure was noticed. They were automatically generated following expert rules over five key variables, namely the oil pressure, the oil temperature, the mean effective torque, the engine RPM, and the engine coolant temperature. The abnormality is here defined as problematic oil pressure behavior. It is worth noting that this highly specific abnormal behavior does not cover all possible abnormal behaviors of the car and thus provides a limited evaluation of our unsupervised anomaly detector.

3. System Model and Problem Formulation

3.1. System Model

The overall system model is represented in Figure 2. We considered a time series T of n samples, each sample being an indexed vector of m dimensions representing variables with $n \in \mathbb{N}$ and $m \in \mathbb{N}$, such that

$$T = \{x_0, x_1, \dots, x_{n-1}\}. \quad (1)$$

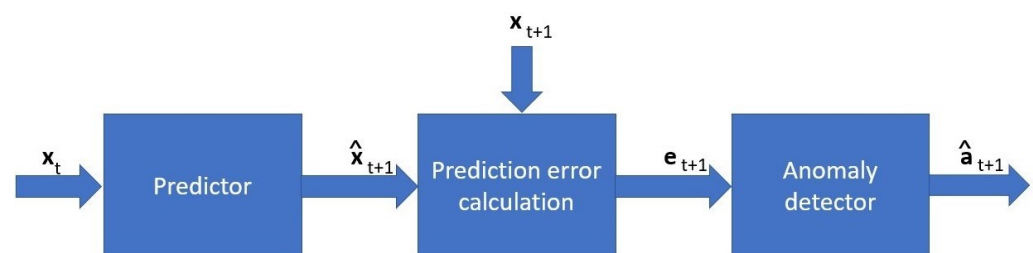


Figure 2. Process of anomaly detection.

Each time series sample $x_t, t \in \{0, \dots, n-1\}$ is potentially corrupted by an anomaly that modifies its “normal behavior” in the sense that the value registered at this point does not fit with the usual pattern of the time series. The goal of the anomaly detector is then to associate each sample x_t with an estimated binary anomaly label $\hat{a}_t \in \{0, 1\}^m$ in order to indicate whether said sample is corrupted by an anomaly. These labels then form the time series A .

$$A = \{\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{n-1}\}. \quad (2)$$

3.2. Unsupervised Anomaly Detection

In this section, we review our baseline algorithm formed by a stacked LSTM-based anomaly detector from References [17,19]. We added the anomaly likelihood method to the anomaly detection process as proposed in Reference [7]. This method allows the anomaly detector to focus on dynamic variations of the prediction error. As different variables at different moments can present behaviors that are more or less difficult to predict, the error prediction average might vary, and this makes the usage of a simple threshold tricky. Anomaly likelihood adapts nicely to this complex behavior, as it evaluates the local probability of a prediction error to occur and also prevents random point anomalies. Finally, we explain below how we acquired all of the parameters.

In Table 1, we have gathered the anomaly detection results found in References [9,13] on the public Yahoo benchmark [32]. We added our LSTM with 50 cells and two layers, as presented in Section 4, using semi-supervised threshold optimization, just as in the work of Reference [10]. For this model, we used 100 epochs, a learning rate of 10^{-3} , and a time window of $w = 100$. We can see that LSTM and CNN performed well compared to some of the more complex models such as generative adversarial networks (TadGAN) or those that are bio-inspired. As we wanted to use small models with simple architectures in order to reduce computational costs while still providing good performance, we chose to exploit LSTM and CNN predictors in this paper. We also chose to compare the GRU [22] with the LSTM, as it is a simplified version of the LSTM. All of these predictors are described below.

3.3. LSTM Predictor

The first block of the anomaly detector is formed by an LSTM neural network, which is an RNN that has proven to be very efficient in capturing the temporal dependencies of a time series using internal memory [15]. The LSTM cell is described in Figure 3. Each cell is recurrent and outputs to vectors c_t and h_t . One LSTM layer can be made of several cells. In the following sections, we use the notation LSTM (n,m) to refer to an LSTM with x cells in the first layer and y cells in the second layer.

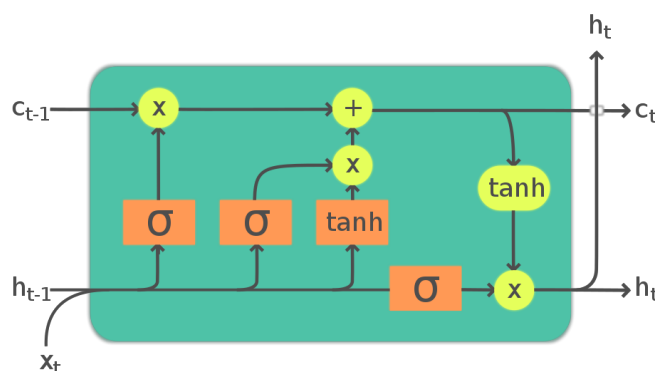


Figure 3. The Long Short Term Memory cell. h_t and c_t are the hidden and context vectors respectively. σ stands for the sigmoid activation function.

The overall stacked LSTM we used has two layers. It takes a sample x_t as an input and predicts the next sample x_{t+1} while using its internal state recurrently. A simple linear layer allows for the conversion of the LSTM output vector h_t into a prediction of x_{t+1} , as shown in Figure 4.

In this paper, we also compare the LSTM with two other models.

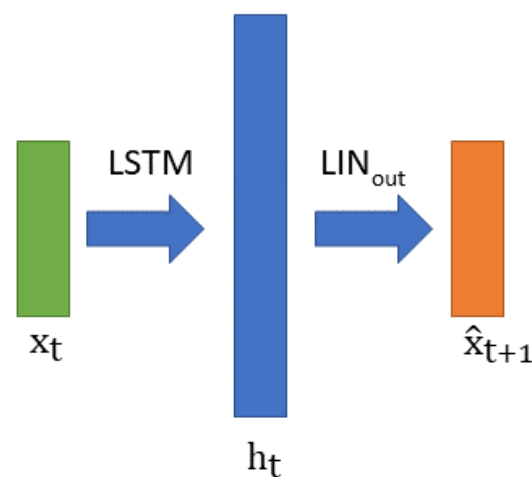


Figure 4. Organization/architecture of the Long Short Term Memory.

3.4. CNN Predictor

A convolutional neural network (CNN) is a well-known machine learning algorithm that has been proven to be very powerful, especially for image classification. In Reference [10], it was used to predict time series and to detect anomalies the same way we proposed in this paper. It is one of the state-of-the-art approaches to unsupervised anomaly detection. The limitation of a CNN lies in its inability to integrate temporal information longer than its convolutional window. As can be seen in Figure 5, the time series is cut into windows in which we run the convolutions. The model can then predict the next time series sample x_{t+w+1} given the window $\{x_t, \dots, x_{t+w}\}$. This CNN model has two layers made up of a 1D convolution layer followed by an ReLU activation function. l_t and h_t are intermediate hidden vectors generated by the model. In the following sections, we use the notation CNN (n,m) to refer to a CNN with n channels in the first layer and m channels in the second layer.

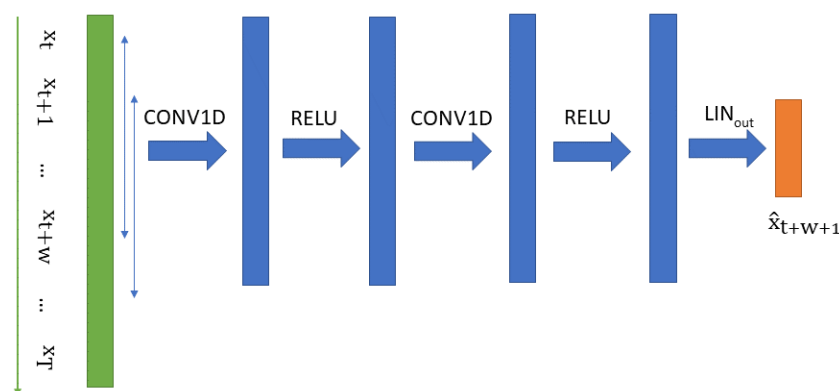


Figure 5. Organization/architecture of the Convolutional Neural Network.

3.5. GRU Predictor

Finally, we propose to use a gated recurrent unit (GRU) network [22], which is a simplified version of the LSTM with fewer gates in each cell forming a lighter RNN. Its architecture is exactly the same as that of the LSTM shown in Figure 4. Depending on the complexity of the task, a GRU can sometimes perform just as well or even better than an LSTM while having fewer parameters. We show such a result very clearly in Table 2. In the following sections, we use the notation GRU (n,m) to refer to a GRU with n cells in the first layer and m cells in the second layer.

Table 2. Performance evaluation comparison between the LSTM [17], GRU, and CNN [10] models. CNN (n,m) refers to a CNN with n channels in the first layer and m channels in the second layer, while LSTM (n,m) and GRU (n,m) refer to an LSTM or GRU with n cells in the first layer and m cells in the second layer.

Model	w_a	f1	TPR	FPR	MSE	MACS	#Parameters
CNN (8-8)	5 s	0.114	0.254	1.66×10^{-4}	0.012	102,240	10,564
CNN (16-16)	5 s	0.114	0.254	1.84×10^{-5}	0.009	211,392	21,812
CNN (32-32)	5 s	0.12	0.253	0	0.007	450,432	46,612
GRU (50-50)	5 s	0.114	0.261	3.12×10^{-5}	0.003	40,600	39,984
GRU (100-100)	5 s	0.114	0.261	2×10^{-5}	0.002	126,200	124,884
GRU (150-150)	5 s	0.114	0.261	0	0.002	256,800	254,784
LSTM (50-50)	5 s	0.114	0.261	1.34×10^{-5}	0.005	52,600	51,884
LSTM (100-100)	5 s	0.114	0.261	2.45×10^{-5}	0.003	165,200	163,684
LSTM (150-150)	5 s	0.114	0.261	2.67×10^{-5}	0.003	337,800	335,484
CNN (8-8)	30 s	0.241	0.251	5.44×10^{-5}	0.012	102,240	10,564
CNN (16-16)	30 s	0.241	0.251	0	0.009	211,392	21,812
CNN (32-32)	30 s	0.247	0.249	0	0.007	450,432	46,612
GRU (50-50)	30 s	0.243	0.258	0	0.003	40,600	39,984
GRU (100-100)	30 s	0.243	0.258	1.31×10^{-5}	0.002	126,200	124,884
GRU (150-150)	30 s	0.243	0.258	0	0.002	256,800	254,784
LSTM (50-50)	30 s	0.243	0.258	1.31×10^{-5}	0.005	52,600	51,884
LSTM (100-100)	30 s	0.243	0.258	0	0.003	165,200	163,684
LSTM (150-150)	30 s	0.243	0.258	0	0.003	337,800	335,484
CNN (8-8)	60 s	0.265	0.249	2.66×10^{-5}	0.012	102,240	10,564
CNN (16-16)	60 s	0.265	0.249	0	0.009	211,392	21,812
CNN (32-32)	60 s	0.268	0.244	0	0.007	450,432	46,612
GRU (50-50)	60 s	0.267	0.254	0	0.003	40,600	39,984
GRU (100-100)	60 s	0.267	0.254	2.58×10^{-5}	0.002	126,200	124,884
GRU (150-150)	60 s	0.267	0.254	0	0.002	256,800	254,784
LSTM (50-50)	60 s	0.266	0.254	2.58×10^{-5}	0.005	52,600	51,884
LSTM (100-100)	60 s	0.266	0.254	0	0.003	165,200	163,684
LSTM (150-150)	60 s	0.266	0.254	0	0.003	337,800	335,484

3.6. Anomaly Detection

During training, for each sample $\mathbf{x}_t$, the predictor output a prediction $\hat{\mathbf{x}}_{t+1}$. The mean square error (MSE) over all dimensions and time was then computed between the predicted and true samples of one time series.

$$\text{MSE} = \frac{1}{nm} \sum_{t=0}^{n-1} ||(\hat{\mathbf{x}}_t - \mathbf{x}_t)^2||_1 \quad (3)$$

where $||\cdot||_1$ stands for the norm 1 of a vector.

This error was used as a loss function in order to optimize the predictor using a gradient descent optimization. Using this method and using an LSTM or a GRU, we were able to take full advantage of having time series of different lengths in the dataset. No

information was lost due to padding or cutting the time series. The predictor was able to efficiently adapt to the variation in time series length.

In a vehicle, most of variables that we exploited here are correlated; for example, a failure in the oil pressure impacts the engine temperature. We made use of this assumption and provided all of the m variables to the predictor together in order to help it to capture more-complex temporal and cross-variable correlations and better predict the signal.

By comparing the true values of the t th sample $\mathbf{x}_t$ with the predicted values, the prediction squared error for the t th sample was formed as:

$$\mathbf{e}_t = (\hat{\mathbf{x}}_t - \mathbf{x}_t)^2 \quad (4)$$

We then obtained the time series of the squared prediction errors:

$$E = \{\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{n-1}\} \quad (5)$$

We then applied the anomaly likelihood method to the time series E as in reference [7] so as to obtain the anomaly scores $\mathbf{s}_t$ and labels $\hat{\mathbf{a}}_t$. Note that we adapted its expression here for the use of multidimensional time series.

First, we computed the parameters μ_t and σ_t^2 of a sliding normal distribution of w samples of E .

$$\mu_t = \frac{\sum_{i=0}^{w-1} \mathbf{e}_{t-i}}{w} \quad (6)$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{w-1} (\mathbf{e}_{t-i} - \mu_t)^2}{w-1} \quad (7)$$

We then used the Gaussian tail probability (Q-function [34]) of the recent sliding average of W' samples of prediction errors in order to obtain the anomaly score.

$$\mathbf{s}_t = 1 - Q\left(\frac{\tilde{\mu}_t - \mu_t}{\sigma_t}\right) \quad (8)$$

where,

$$\tilde{\mu}_t = \frac{\sum_{i=0}^{W'-1} \mathbf{e}_{t-i}}{W'} \quad (9)$$

Finally, we used a threshold vector $\boldsymbol{\theta}$ to compute the found anomalies.

$$\hat{\mathbf{a}}_t^i = \begin{cases} 1 & \text{if } \mathbf{s}_t^i \geq \theta^i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $\hat{\mathbf{a}}_t^i$ is the scalar of the i th dimension of the vector $\hat{\mathbf{a}}_t$.

After the prediction, each variable was treated separately with its own parameters; thus, there were m anomaly likelihoods being processed in parallel.

We finally defined the set A_T of timestamps in which at least one variable was found to be abnormal.

$$A_T = \{t \in [t_0, t_{n-1}] \mid \|\hat{\mathbf{a}}_t\|_1 > 0\} \quad (11)$$

With t_i being the timestamp of the corresponding time series sample.

4. Results

After learning the predictor parameters, we ran our unsupervised anomaly detector through all the time series of our dataset and then exploited the results through different evaluations.

For all models, during the learning phase we used 300 epochs, a learning rate of 10^{-4} with the Adam optimization, and 2000 subsequences of 100 points each; additionally, we used 20% of the data for testing. The Adam optimization is one of the best and most-utilized

gradient-descent-based optimization algorithms for neural networks such as CNN and RNN that exists in the field of machine learning [35].

Below, we will first evaluate how well our algorithm catches the given anomalies using our specific labels, and then we will propose measuring the correlation between labels and variables using the multidimensional nature of our algorithm.

4.1. Evaluation with Labels (Oil Pressure Failure)

As explained in Section 2, labels that have been provided to our team express a specific abnormal behavior due to failures in oil pressure and do not cover all possible abnormal behaviors.

These labels point to specific timestamps in the dataset where this oil pressure failure has been detected. This abnormal behavior is not instantaneous and appears for several seconds. Thus, we have to consider a time window w_a around those timestamps' labels to assess whether an abnormal sample found by our algorithm matches one label. We define the set of true positives TP as:

$$TP = \{t \in A_T | \exists t_l \in L, t_l - w_a \leq t \leq t_l + w_a\} \quad (12)$$

With L and A_T being, respectively, the set of labels and found anomalies timestamps.

We only consider one true positive sample with respect to one label even if there are several true positives in that label window, hence the usage of the set formalism.

We then define the false positive (FP), true negative (TN) and false negative (FN) sets as follows.

$$FP = \{t \in A_T \mid \nexists t_l \in L, t_l - w_a \leq t \leq t_l + w_a\} \quad (13)$$

$$TN = \left\{ t \in [t_O, t_{n-1}] \setminus A_T \mid \nexists t_l \in L, t_l - w_a \leq t \leq t_l + w_a \right\} \quad (14)$$

$$FN = \left\{ t \in [t_O, t_{n-1}] \setminus A_T \mid \exists t_l \in L, t_l - w_a \leq t \leq t_l + w_a \right\} \quad (15)$$

In order to obtain a clearer evaluation, we then compute respective rates—namely the true positive rate (TPR) for TP and so on—corresponding to those last ensembles as:

$$TPR = \frac{|TP|}{|L|} \quad (16)$$

$$FPR = \frac{|FP|}{n} \quad (17)$$

$$TNR = \frac{|TN|}{n} \quad (18)$$

$$FNR = \frac{|FN|}{n} \quad (19)$$

Provided in Table 3 are the TPR, FPR, TNR, and FNR obtained with windows w_a of 5, 30, and 60 s and an LSTM with two layers of 50 cells with the notation LSTM (50-50).

Table 3. Performance evaluation of the LSTM (50-50) on limited labels.

w_a	TPR	FPR	TNR	FNR
5 s	0.19	0.065	0.91	0.0093
30 s	0.62	0.062	0.079	0.0082
60 s	0.86	0.061	0.64	0.0076

The best results were obtained using the largest $w_a = 60$ s, and they achieved 86% successful detection.

The FPR is 0.061%. If false positives were normally distributed over time and considering our 1 Hz sampling rate, this would mean that the model sends a false positive alert, on average, every 16 s. First, we have to remember here that our labels are limited to one specific kind of anomaly, and thus, this relatively high FPR might not depict an unsatisfactory behavior of our unsupervised anomaly detector. A subset of anomalies might correspond to actual abnormal behaviors of the vehicle which are not given in our labels.

The evaluation of whether our false positives match abnormal behaviors is not always straightforward. Visual interpretation can be difficult, as the model processes 85 variables altogether.

In Figure 6, we show an example of a successful detection made by our model regarding the oil pressure problem. As the engine mean effective torque is rising, the oil pressure should do the same; however, it does not do so here, which constitutes abnormal behavior. We also see how anomaly likelihood offers a nice dynamic adaptation of the quadratic error processing to find anomalies.



Figure 6. Blue: signal, Results for a slice of the engine oil pressure (a) and engine mean effective torque (b) time series. On the left side, we see a specific abnormal behavior of the oil pressure. As the mean effective torque rises, the oil pressure is supposed to do so as well, but it does not do so here, which constitutes an abnormal behavior that our model successfully highlighted despite a label is missing at this timestep. On the right side, we see an example of a true positive and the impact of the anomaly likelihood, which adapts to the variation of the prediction error from the left to the right side. This method also induces a little delay in the anomaly detection time because of the local average that is computed.

4.2. LSTM Computation Costs Comparison

Here we evaluate LSTM models with respect to different amounts of cells and layers. We start with our baseline LSTM (50-50) and run smaller architectures. We also compute the number of parameters and multiply–accumulate operations (MACs) for each model in

order to evaluate their computational costs using the pytorch-OpCounter (access date: 12 January 2022) library <https://github.com/Lyken17/pytorch-OpCounter>.

In Table 4, the LSTM (10) shows the best TPR, but, for a better evaluation, the FPR has to be considered, as discussed below.

Table 4. Performance evaluation comparison between different Long Short Term Memories for $w_a = 60$ s.

Model	TPR	FPR	TNR	FNR
LSTM (10)	0.86	0.061	0.64	0.0076
LSTM (50)	0.68	0.048	0.65	0.0081
LSTM (10-10)	0.84	0.068	0.63	0.008
LSTM (50-50)	0.69	0.049	0.65	0.008

When evaluating the anomaly detection results, the two main factors to be considered are the TPR and the FPR. We want the highest TPR while keeping the FPR as low as possible. Depending on the threshold used during anomaly detection, TPR and FPR vary. It is difficult to equalize at least one of these across different models. Thus, we propose to use the positive likelihood ratio (PLR) defined in Equation (20), which allows for better evaluation and comparison between different models.

$$PLR = \frac{TPR}{FPR} \quad (20)$$

In Table 5, we first see that the LSTM (50) gives the best PLR of 14.2. We can also observe that the LSTM (10) with only 10 cells performs very similarly with a PLR of 14.1 but offers a computation cost about 6.6 times lower. As a result, our best predictor for anomaly detection is the LSTM (10). This is one of the lightest in terms of parameters and MACs, which makes it a better candidate for embedded systems. However, it is also one of the worst predictors with an MSE of 0.0059. This is an interesting property of the overall anomaly detector process that allows for the use of simpler predictors despite the prediction performance limitation.

Table 5. Computation costs comparison between different LSTMs; the PLR is computed with respect to $w_a = 60$ s.

Model	Test MSE	PLR	MACS	#Parameters
LSTM (10)	0.0059	14.1	4.8 K	4.8 K
LSTM (50)	0.0021	14.2	31.8 K	31.5 K
LSTM (10-10)	0.007	12.4	5.7 K	5.6 K
LSTM (50-50)	0.002	14	52.6 K	51.9 K

In order to evaluate more directly and quantitatively each state-of-the-art model, including the CNN [10], we have tested all models with various sizes and without the anomaly likelihood. The method used to find anomalies this time is semi-supervised, and we also use the f1 score defined in Equation (23) to evaluate our results [10]. The model is processed exactly as before for training and prediction.

$$\text{recall} = \frac{TP}{P} \quad (21)$$

where TP is the number of true positives and P the number of real positives.

$$\text{precision} = \frac{TP}{PP} \quad (22)$$

where PP is the number of predicted positives.

$$f1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (23)$$

Instead of using the anomaly likelihood method, we simply search for the anomaly threshold that maximizes the PLR. This threshold is the value above which the prediction error is considered to be abnormal. As finding this threshold requires labels, this method is therefore semi-supervised. This allows us to focus our performance analysis more on the ML model itself and avoid unsupervised post-processing bias.

In Table 2, all of the models' performances are given in detail. We can see that the best f1 score was achieved by the CNN (32-32) followed very closely by the GRU (50-50) which is much lighter in terms of computational costs.

Considering the window $w_a = 5$ s for example, we observe that the CNN (8-8), the GRU (50,50), and the LSTM (50,50) reach the same f1. However, this GRU has 23% reduced parameters and consumes 23% fewer MACs than this LSTM. This GRU, while having 3.79 times greater parameters than this CNN, still consumes 60% fewer MACs than this CNN. This result highlights that CNN might not always represent the best solution for time series, as convolutions induce a lot of calculation over a given time window. As no information is integrated from one time window to another, a CNN indeed needs to process N sliding time windows while an LSTM or a GRU processes N samples.

In terms of TPR and FPR, the GRU also shows the highest performance with the lowest computational costs. From one time window w_a to another, the signal is being resampled in our implementation method, meaning that signals do not have the same length and some labels can be merged together. This has an impact over the range of f1, TPR, and FPR scalars, which makes comparisons between those metrics across time windows difficult.

In accordance with the previous results, we see that the f1 score, TPR, and FPR are not or very slightly impacted by the model size whereas the prediction performance increases accordingly to with size. We thus show again that the best prediction model might not be the best anomaly detection model in terms of anomaly detection performance and computational costs. Finally, the proposed semi-supervised anomaly threshold optimization method offers very low or null FPRs.

In the next section, we present a method to correlate variables with a specific kind of anomaly.

4.3. Correlation of Variables with Labels

As our labels are focused on one specific abnormal behavior, the question was raised as to whether it is possible to explain which variables are linked to that behavior.

In our case, abnormal behavior has been defined by experts using five expert variables: MeanEffTorque, EngineRPM, RST_EngineOilPressure, RST_EngineCoolantTemp, and RST_OilTemperature. The anomaly is linked to a problem with the oil pressure.

Making use of our anomaly detector and the way it handles each variable independently after prediction, we propose to use our model PLR to correlate variables with the anomaly, the underlying idea being that variables showing better detection are more linked to the anomaly.

We look at variables with the highest PLR. In Figure 7, we see that three of our five expert variables are ranked within the first quarter of variables with the best PLR out of the total 85 variables. Modifying the parameters of the anomaly detection model will impact this result, but we always find most of the five expert variables in that best quarter. Please note that we find oil pressure to be the second-best variable. This method thus effectively points out variables that are correlated with the anomaly, although we would need additional expert feedback in order to understand the role of the neighbor variables. We also want to know whether they are correlated to the anomaly. Unfortunately, we were not able to obtain that feedback yet.

EngineAirTemp	2.348447
LongitudinalAccelerationProc	2.356447
ExternalTemp	2.365422
RST_IntakeAirTemperature	2.414615
Y	2.417719
EngineRPM	2.438932
MeanEffTorque	2.454392
Z	2.455398
FrontRightBrakeTemperature	2.472738
Longitude	2.494319
ASRStaticTorqueRequest	2.516906
RequestedTorqueAfterProc	2.557454
ASRDynamicTorqueRequest	2.580514
RawEngineTorque_WithoutTMReq	2.598110
RST_BoostPressure_V2	2.644864
RearWheelAnglePosition	2.725587
YawRate_Raw	2.727440
ComputedOilDrainingRange	2.730209
Latitude	2.735888
NominalFuelConsumption	2.801802
FuelConsumption	2.817450
NumSats	2.851533
RST_EngineOilPressure	2.871740
Altitude	2.953266

Figure 7. Variables correlations with labels. Variables are shown ordered by decreasing PLR. Circled in green are the variables that are known by experts to have caused the oil pressure failure.

5. Discussion and Future Work

In this paper, we have proposed unsupervised anomaly detection models using different state-of-the-art algorithms such as a LSTM, GRU, or CNN and anomaly likelihood or a semi-supervised anomaly threshold optimization. We applied these models in order to effectively point out abnormal temporal behaviors over a real, complex, multivariate sensor time series that was extracted from a vehicle CAN bus during circuit tests.

The evaluation of these models with respect to some limited labels shows an acceptable rate of detection, although there is still room for improvement. First, as our labels only express one specific abnormal behavior, it is difficult to fully evaluate our model. Second, further advanced data preprocessing and feature engineering conducted with experts would certainly help.

The comparison between different LSTMs, GRUs, and CNNs shows that the anomaly detector can work just as well or even better with smaller predictors despite a slight decrease prediction performance. Under certain circumstances, we have shown that the GRU can save up to 23% of parameters and 60% of MACs operations compared to the LSTM and the CNN. This is an interesting property for embedded systems, as it allows for a reduction of the algorithm computational cost.

As future works, we intend to embed anomaly detectors inside vehicles and thus expose our models directly to high computational cost constraints. We are currently studying spiking neural networks as a candidate to tackle this problem. An in-depth anomaly detection performance and computational costs study should be performed in the future on public data benchmarks so as to focus on the reduction of computational costs, as this is lacking in the literature presented in this paper.

Finally, we proposed a method for evaluating the correlation between variables and a specific abnormal behavior given in labels. This method shows interesting results and would need further in-depth evaluation in order to refine it.

Author Contributions: Conceptualization, Y.C., A.V., B.M. and A.P.; methodology, Y.C., A.V., B.M. and A.P.; software, Y.C.; validation, A.V., B.M. and A.P.; investigation, Y.C. and A.V.; data curation, Y.C. and A.V.; writing—original draft preparation, Y.C. and A.P.; writing—review and editing, Y.C. and A.P.; supervision, B.M., A.V. and A.P.; project administration, A.P.; funding acquisition, B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Renault and the ANRT (Research contract LEAT—Renault n°2021-C-5854/CNRS n°239386).

Data Availability Statement: The Yahoo anomaly benchmark can be found here: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70> (accessed on 12 January 2022). The Renault car data is publicly unavailable.

Acknowledgments: This material is based upon work supported by the French technological research agency (ANRT) through a CIFRE thesis in collaboration with Renault.

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

References

- Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 15. [\[CrossRef\]](#)
- Shaukat, K.; Alam, T.M.; Luo, S.; Shabbir, S.; Hameed, I.A.; Li, J.; Abbas, S.K.; Javed, U. A review of time-series anomaly detection techniques: A step to future perspectives. In *Advances in Information and Communication, Proceedings of the 2021 Future of Information and Communication Conference (FICC), Vancouver, BC, Canada, 29–30 April 2021*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 1, pp. 865–877.
- Guha, S.; Mishra, N.; Roy, G.; Schrijvers, O. Robust random cut forest based anomaly detection on streams. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
- Kejariwal, A. Introducing Practical and Robust Anomaly Detection in a Time Series. Twitter Engineering Blog. Web, 15. 2015. Available online: https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series (accessed on 12 January 2022).
- Stanway, A. Etsy Skyline. Online Code Repos. 2013. Available online: <https://github.com/etsy/skyline> (accessed on 12 January 2022).
- Laptev, N.; Amizadeh, S.; Flint, I. Generic and scalable framework for automated time-series anomaly detection. In Proceedings of the ACM SIGKDD International Conference, Sydney, NSW, Australia, 10–13 August 2015; pp. 1939–1947.
- Ahmad, S.; Lavin, A.; Purdy, S.; Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [\[CrossRef\]](#)
- Widanage, C.; Li, J.; Tyagi, S.; Teja, R.; Peng, B.; Kamburugamuve, S.; Baum, D.; Smith, D.; Qiu, J.; Koskey, J. Anomaly detection over streaming data: Indy500 case study. In Proceedings of the 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 8–13 July 2019; pp. 9–16.
- Maciag, P.S.; Kryszkiewicz, M.; Bembenik, R.; Lobo, J.L.; Del Ser, J. Unsupervised anomaly detection in stream data with online evolving spiking neural networks. *Neural Netw.* **2021**, *139*, 118–139. [\[CrossRef\]](#) [\[PubMed\]](#)
- Munir, M.; Siddiqui, S.; Dengel, A.; Ahmed, S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access* **2019**, *7*, 1991–2005. [\[CrossRef\]](#)
- Ren, H.; Xu, B.; Wang, Y.; Yi, C.; Huang, C.; Kou, X.; Xing, T.; Yang, M.; Tong, J.; Zhang, Q. Time-series anomaly detection service at microsoft. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 3009–3017.
- Zhao, H.; Wang, Y.; Duan, J.; Huang, C.; Cao, D.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; Zhang, Q. Multivariate time-series anomaly detection via graph attention network. In Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 841–850.
- Geiger, A.; Liu, D.; Alnegheimish, S.; Cuesta-Infante, A.; Veeramachaneni, K. Tadgan: Time series anomaly detection using generative adversarial networks. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 33–43.
- Långkvist, M.; Karlsson, L.; Loutfi, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognit. Lett.* **2014**, *42*, 11–24. [\[CrossRef\]](#)
- Hochreiter, S.; Schmidhuber, J. *Long Short-Term Memory*; MIT Press: Cambridge, MA, USA, 1997; Volume 9, pp. 1735–1780.
- Bontemps, L.; McDermott, J.; Le-Khac, N. Collective anomaly detection based on long short-term memory recurrent neural networks. In Proceedings of the International Conference of Future Data and Security Engineering, Can Tho City, Vietnam, 23–25 November 2016.
- Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long Short Term Memory Networks for Anomaly Detection in Time Series. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 22–24 April 2015.
- Filonov, P.; Lavrentyev, A.; Vorontsov, A. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *arXiv* **2016**, arXiv:1612.06676.
- Chauhan, S.; Vig, L. Anomaly detection in ECG time signals via deep long short-term memory networks. In Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–21 October 2015.
- Cherdo, Y.; De Kerret, P.; Pawlak, R. Training lstm for unsupervised anomaly detection without a priori knowledge. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 4297–4301.

21. Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on KNOWLEDGE Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 387–395.
22. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
23. Meyer, P.; Häckel, T.; Korf, F.; Schmidt, T.C. Network anomaly detection in cars based on time-sensitive ingress control. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, 18 November–16 December 2020; pp. 1–5.
24. Rajbahadur, G.K.; Malton, A.J.; Walenstein, A.; Hassan, A.E. A survey of anomaly detection for connected vehicle cybersecurity and safety. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 421–426.
25. 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. [[CrossRef](#)]
26. Sun, H.; Chen, M.; Weng, J.; Liu, Z.; Geng, G. Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10880–10893. [[CrossRef](#)]
27. Boumiza, S.; Braham, R. An anomaly detector for CAN bus networks in autonomous cars based on neural networks. In Proceedings of the 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 21–23 October 2019; pp. 1–6.
28. Wang, Y.; Masoud, N.; Khojandi, A. Real-time sensor anomaly detection and recovery in connected automated vehicle sensors. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 1411–1421. [[CrossRef](#)]
29. Narayanan, S.N.; Mittal, S.; Joshi, A. OBD_SecureAlert: An anomaly detection system for vehicles. In Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 18–20 May 2016; pp. 1–6.
30. Bogdoll, D.; Nitsche, M.; Zöllner, J.M. Anomaly detection in autonomous driving: A survey. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 4488–4499.
31. Pereira, P.J.; Coelho, G.; Ribeiro, A.; Matos, L.M.; Nunes, E.C.; Ferreira, A.; Pilastrri, A.; Cortez, P. Using deep autoencoders for in-vehicle audio anomaly detection. *Procedia Comput. Sci.* **2021**, *192*, 298–307. [[CrossRef](#)]
32. Yahoo! Webscope Research. S5—A Labeled Anomaly Detection Dataset, Version 1.0(16M). Available online: <https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70> (accessed on 12 January 2022).
33. Lavin, A.; Ahmad, S. Evaluating Real-Time Anomaly Detection Algorithms—The Numenta Anomaly Benchmark. In Proceedings of the 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 9–11 December 2015; pp. 38–44.
34. Karagiannidis, G.K.; Lioumpas, A.S. An improved approximation for the Gaussian Q-function. *IEEE Commun. Lett.* **2007**, *11*, 644–646. [[CrossRef](#)]
35. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.