

Received 14 September 2022, accepted 25 September 2022, date of publication 3 October 2022, date of current version 10 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3211387

RESEARCH ARTICLE

Adaptive Ensemble Methods for Tampering Detection in Automotive Aftertreatment Systems

ROLAND BOLBOACĂ 

The Faculty of Engineering and Information Technology, George Emil Palade University of Medicine, Pharmacy, Science, and Technology of Târgu Mureș, 540139 Târgu Mureș, Romania

e-mail: roland.bolboaca@umfst.ro


This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the Smart Adaptive Remote Diagnostic Anti-Tampering Systems (DIAS) Project (<https://dias-project.com/>) under Grant 814951. This document reflects only the author's view and the Agency is not responsible for any use that may be made of the information it contains.

ABSTRACT Control and diagnostic processes in modern vehicles incorporate nowadays a wide set of functionalities to preserve the vehicle's health. Automotive vehicles contain embedded systems that must perform a diverse palette of tasks, ranging from less critical tasks (e.g., audio/video media control), to crucial ones, such as controlling the engine, fuel consumption, or the aftertreatment system. This paper identifies and addresses one emerging threat, namely, automotive tampering. Tampering denotes a procedure that changes the behavior of the system to gain financial or functional advantages, without damaging the system and without triggering the built-in safety features of the vehicle. Numerous studies show a growing number of tampered vehicles worldwide and considering that tampered vehicles contribute to air and atmosphere pollution, tampering remains a serious environmental threat. This paper proposes two ensemble-based approaches for tampering detection, both using Long Short-Term Memory neural network predictors, together with Cumulative Sum and Histogram distance-based detectors. Additionally, an Adaptive Majority Weighted Voting fusion methodology is proposed, that considers the historical decisions of the detectors. Experimental results are based on three unique datasets that incorporate a multitude of tampering scenarios. The results prove the efficiency of the proposed ensembles, with a 0% false alert rate and up to 100% detection rate, even when dealing with intelligent tamperers, and even in comparison with state-of-the-art tampering detection solutions. Moreover, this paper offers resource consumption and scalability measurements on a reference embedded system, further demonstrating the integrability of the proposed techniques in a real embedded environment.

INDEX TERMS Automotive, anomaly detection, tampering detection, ensemble, outlier detection, exhaust aftertreatment system, long short-term memory, histogram distance, cumulative sum, teacher forcing.

I. INTRODUCTION

In recent years, a handful of studies focused on detecting, and preventing cyber-attacks in automotive systems, ranging from anomaly detection techniques [1], [2], firewalls [3], [4], intrusion detection systems [3], [5], [6], to cryptography-based methods for assuring authenticity and confidentiality of data frames [7], [8], [9]. While the lack of security for the onboard systems is alarming, the advancement in Vehicle-to-Everything (V2X) and Vehicle-to-Vehicle (V2V) technologies are connecting vehicles to the cloud and to other vehicles, in consequence opening the vehicle networks

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine .

to a new plethora of threats, vulnerabilities, and attack surfaces [10], [11].

One specific emerging threat in recent years is vehicle *tampering*. The distinction between tampering and cyber-attacks, is that tampering denotes a procedure that changes the behavior of the system to gain a specific advantage or profit. Thus, it is important to emphasize that the *tamperer* (i.e., the one who performs tampering) does not intend to cause damage to the system, but to gain a financial or functional advantage by manipulating the system. Vehicle tampering can take many forms, from modifying the readings of an odometer [12] to altering the Emission Control System (ECS) [13]. Advanced tampering currently ranges from altering or disabling certain sub-systems (e.g., aftertreatment), to signal manipulation,

code injection or reflashing of control units, to actual concealing of the tampering devices [14], [15]. As a response, the Horizon2020 project Smart Adaptive Remote Diagnostic Anti-Tampering Systems (DIAS) emerged, to research and develop methods for hardening the vehicle Environmental Protection System (EPS) against known and unknown (e.g., possible future) tampering methods. The main objectives of the project include in-vehicle detection methods, as well as counter-measure solutions, guidelines and recommendations for future anti-tampering legislation, with the final goal of partial or total elimination of automotive tampering attempts [16].

The European Commission (EC) estimates that up to 50% of second-hand vehicles sold across the European Union (EU) were subjected to odometer tampering [17], while a study from the Danish Ministry of Environment and Food reveals that up to 25% of heavy-duty vehicles in Denmark may be tampered [18]. Giechaskiel et al. [19] revealed the true magnitude of tampering with reports from 2017. The authors point out that a large number of sites, 87 to be exact, were selling tampering devices for Euro IV - VI vehicles, all across EU. Furthermore, the authors point out that up to 25% of vehicles have tampering devices. More alarming is the fact that in Europe there is no clear anti-tampering legislation in place, compared to the United States of America, where tampering is prohibited by law. Even so, in 2020 the Air Enforcement Division of the United States Environmental Protection Agency presented a study where more than half a million pickup trucks had their emission control tampered, this corresponding to 15% of Class 2b and 3 diesel trucks produced after 2003 [20]. Additionally, the study addresses automotive tampering in Canada, revealing that in 2007 at least 20% of light-duty vehicles were responsible for almost 80% of the emissions, while out of 6000 heavy-duty vehicles analyzed, 26% of them were high emitters. Considering the fact that tampered vehicles contribute to air and atmosphere pollution, while altered ECS produces excess emissions of nitrogen oxides (NO_x), particulate matter (PM), and other pollutants, tampering still remains an unresolved threat for the environment, and for human health as well.

The complexity of the problem at hand requires a suitable solution. Therefore, this paper proposes two tampering detection ensembles positioned at the application level as independent Electronic Control Unit (ECU) applications. The proposed ensembles leverage the predictive capabilities of Long Short-Term Memory Neuronal Network (LSTM-NN), which are used as predictors, together with Cumulative Sum (CUSUM) chart and Histogram distance-based detection approaches. The LSTM-NN are empowered with a Teacher Forcing (TF) [21] variant that allows to feed back, at the current time-step, the previously observed values of the output monitored variables, in parallel with the set of inputs, during both the training and testing phase. Thus, an open-loop forecasting methodology is proposed, resembling a Series-Parallel architecture of Nonlinear Auto-Regressive Neural Networks with exogenous inputs (NARXNN) [22].

The usage of Teacher Forcing (TF) represents a creative way of applying TF to detection techniques (e.g., anomaly detection, tampering detection). Moreover, the proposed ensembles use an Adaptive Majority Weighted Voting (AMWV) fusion methodology, that takes into account the historical decisions of each detector, and outputs one of three decisions: normal, alert and warning. Lastly, this paper showcases an offline training methodology for predictors, to reduce the complexity of the model training procedures.

The proposed tampering detection solutions were tested on three distinct datasets and were validated using eight performance metrics on different tampering types and driving scenarios. The first dataset was produced by a state-of-the-art aftertreatment simulation model of a heavy-duty vehicle developed at the Laboratory of Applied Thermodynamics from Aristotle University of Thessaloniki. The second dataset was created by the Vehicle Emissions Heavy Duty chassis laboratory at the Joint Research Centre of the EC, and the third dataset originates from a Skoda Rapid passenger vehicle. The proposed solutions obtained notable results, including 0% False Positive Rates on all datasets and up to 100% detection rates in most of the cases. Furthermore, the ensembles were compared to state-of-the-art tampering detection methodologies [23], [24], with promising results. Additionally, the comprehensive experimental assessment conducted on real and simulated tampering scenarios, intends to push forward towards a more resilient, generalized tampering detection solution. Finally, in addition to the previously mentioned, this paper provides resource consumption and scalability measurements on a reference embedded system, demonstrating the possibility of integrating the proposed solutions in an actual embedded environment.

The remainder of the paper is organized as follows. A description of the threat model and Automotive Exhaust Aftertreatment System (EAS) tampering is offered in Section II. The proposed tampering detection solutions are presented in Section III. Next, the datasets are described, together with the final design of the proposed ensembles in Section IV. Section V presents the experimental results, following a series of discussions in Section VI. Afterwards, the most relevant related studies are presented in Section VII. The paper concludes in Section VII.

II. THREAT MODEL AND AUTOMOTIVE EAS TAMPERING

A. THREAT MODEL

The considered threat model assumes that a tamperer (e.g., attacker) is capable of altering the vehicle's sub-systems without causing physical damage or affecting its safety. A tamperer is assumed to have unrestricted physical access and unlimited time to add, change or remove certain vehicle components (e.g., remove/replace sensors). It is further assumed that the tamperer has access to the in-vehicle networks. Thus, this assumed tamperer is capable of injecting messages and emulating certain signals (e.g., via emulators), while remaining stealthy (i.e., without causing alerts to be

generated by the on-board diagnostic systems), or that the tamperer is capable of deleting any diagnostic trouble codes that might result from tampering (e.g., by connecting Diagnostic Trouble Code erasers to the vehicle's network). It is also assumed that tampering is performed with the vehicle's owner consent or that the vehicle owner is the one performing the tampering. Tampering is assumed to be persistent over an extended period of time. The assumed scope of tampering is for the vehicle's owner to gain certain advantages (e.g., operational, financial). Lastly, it is considered that the environment in which the ensembles run is protected via state-of-the-art security solutions (e.g., code signing, secure boot).

B. AUTOMOTIVE EXHAUST AFTERTREATMENT SYSTEM TAMPERING

Developed as a response to the constantly increasing vehicle emissions, the Exhaust Aftertreatment System (EAS) reduces the pollutants generated by diesel engines by converting them to less harmful elements. Early EAS models were equipped with metal catalysts which oxidized carbon monoxides and hydrocarbons, reducing them to carbon dioxide and water. Modern EAS models utilize additional catalytic converters, sensors, and automated processes to achieve, not only higher efficiency in reducing pollutants but also reduced production and maintenance costs [25]. Pollutants, such as carbon monoxide (CO), hydrocarbons (HC), nitrogen oxides (NO_x) and particulate matter (PM) after passing through the EAS are converted to less harmful elements like carbon dioxide (CO_2), water (H_2O) and nitrogen (N_2). Pollutant reduction operations are performed by the catalytic converters including Oxidation catalysts (oxycats), Three-way catalysts (TWCs), Lean NO_x traps (LNTs), Particulate Filters (PFs), Selective Catalytic Reduction (SCR) systems and Ammonia Slip Catalyst (ASC) [25].

Advancements in the EAS design, in conjunction with stricter regulations imposed by governments, led to a considerable reduction in emission levels. However, manipulations of the EAS are still discovered during periodical vehicle inspections [14]. Emulators used on heavy-duty vehicles affect the functionality of the SCR systems and the NO_x sensors. Here, the motivation for tampering remains a financial one, to save funds on Diesel Exhaust Fluid (DEF) (e.g., AdBlue) and maintenance. DEF is utilized by the SCR in the NO_x reduction process. Disabling the dosing of DEF causes increased NO_x emissions, which are hidden by injecting false NO_x values using the emulators.

For the reader to get a better understanding of how efficient this kind of emulators can be, Figure 1, illustrates the downstream NO_x readings in the presence and absence of an emulator. Here, the measured NO_x values when DEF is normally dosed are illustrated in the top sub-figure. The measured NO_x values when DEF dosing is reduced (e.g., tampered) are illustrated in the middle sub-figure. Finally, the false NO_x values injected by an emulator are illustrated in the bottom sub-figure. As shown, the injected false values closely match the real readings while the average real NO_x

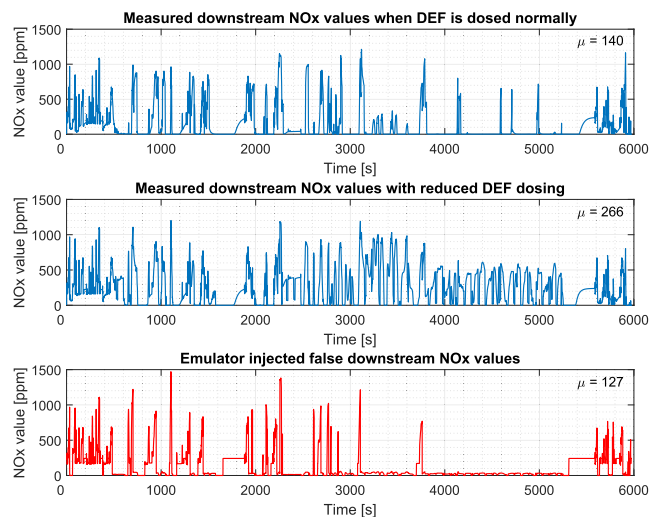


FIGURE 1. Illustration of the NO_x concentration measurements with normal/reduced DEF dosing vs the false NO_x values injected by an emulator.

concentration values doubles. The cumulative values for the NO_x measurements increase from $8 * 10^6$ ppm to $16 * 10^6$ ppm for a driving cycle of 6000 seconds.

Nowadays, tampering is supplied in specialized vehicle workshops both as a service and as product. For the latter, instructions are provided through online shops, forums, and even social media websites [19], making it easily accessible for everyone interested in it. Furthermore, Giechaskiel *et al.* [19] published data related to the real magnitude of tampering. The study shows that the NO_x emissions of tampered passenger vehicles can have NO_x values increased up to 850 times. On the other hand, Euro IV heavy-duty trucks can have the NO_x values increased up to 220 times. For a Non-Road Mobile Machinery (NRMM), the results show an increase of NO_x emissions of over 200 times. Concurrently, [20] brings forward research data showing that the emissions of tampered vehicles are increasing from 30 to 300 times in terms of NO_x values, by three orders of magnitude in terms of Non-methane hydrocarbons, by two orders of magnitude in terms of carbon monoxide values, and by 15 to 40 times in terms of particle matter.

These issues become worse when not only single vehicles, but entire fleets are tampered, for extended periods of time. As previously mentioned, emulators can be easily purchased online, at low prices and often times they can be installed with ease just by connecting them to the On-board diagnostics (OBD) port. This, however, is only one tampering method, other more advanced methods require additional hardware/software manipulations together with the use of more advanced emulators.

III. PROPOSED TAMPERING DETECTION SOLUTIONS

The two proposed tampering detection ensembles, further denoted as Cumulative Sum Based Ensemble (CBE) and Histogram Based Ensemble (HBE), both use predictive models

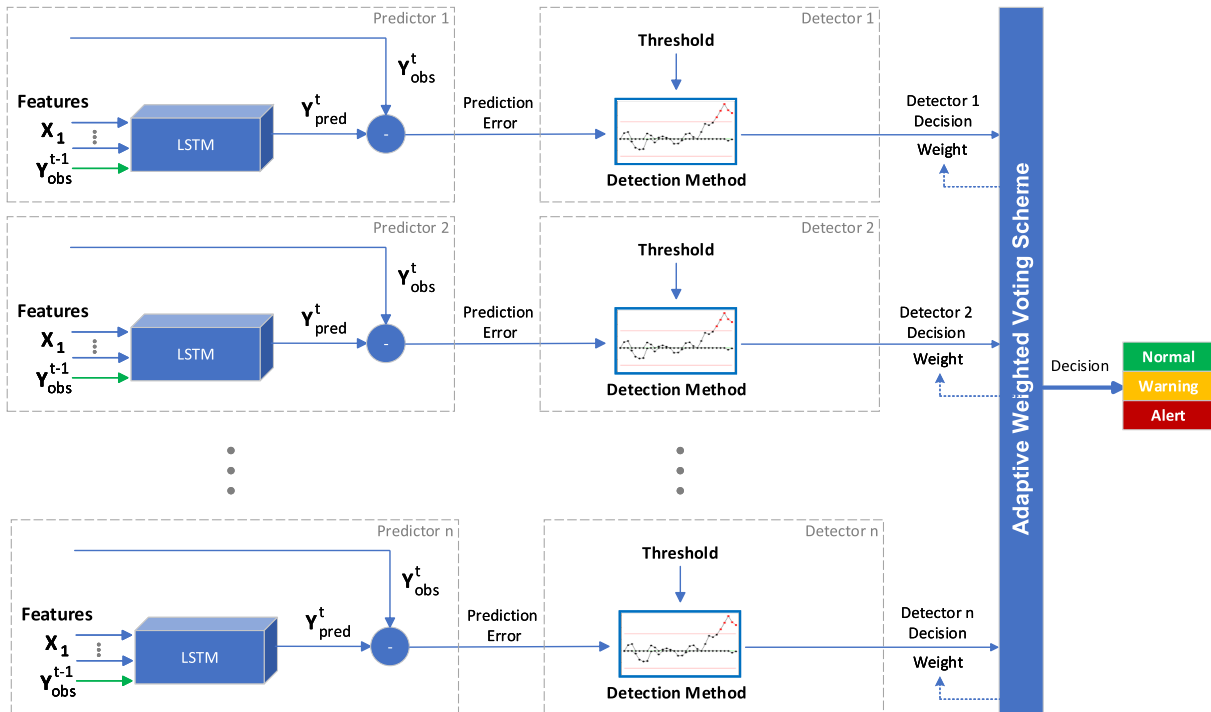


FIGURE 2. General overview of the proposed ensembles architecture. The two proposed ensembles are differentiated by the detection method used by the base detectors (e.g., CUSUM, Histogram distance).

and detectors. Predictive models have long been used for anomaly detection, and more recently for tampering detection [23]. Compared to statistical profiling and clustering methods [26], [27], [28], [29], predictive models are built using historical data from which they *learn* the trends and seasonality of the time-series data. While statistical profiling and clustering methods are suitable for detecting obvious anomalies, represented by points or series of points which deviate significantly from the baseline, in the case of tampering, the anomalies are subtle and almost identical to the normal observations, as depicted in Figure 1.

The proposed ensembles are envisioned to function as in-vehicle solutions, at application level in the Electronic Control Units (ECUs). The distancing from the network level has several advantages, for instance, the ability to function on top of different communication protocols (e.g., LIN, CAN, Flex-Ray, Ethernet), without requiring information about the data frame structure or frequency.

One crucial aspect of the proposed techniques is the offline training. The processing capabilities of the ECUs are extremely limited, while the training procedures of the base predictors (e.g., LSTM-NNs) are slow and demand high computational resources. Nonetheless, a solution to this problem is an offline training methodology. Predictors are trained and tested outside the vehicle with tamper-free measurements, and a variant of Teacher Forcing (TF), where the ground truth value from the current time-step is used as input to the next time-step. Furthermore, the detection thresholds are also computed in an offline manner. The trained models (i.e., the

weight matrices of the LSTM-NNs, the hyper-parameters and the thresholds) are later deployed onto the ECUs.

During the detection phase, each detector from the ensembles monitors a signal (e.g., a variable) by analyzing the deviations from the normal learned behavior.

A. ENSEMBLE ARCHITECTURE

A common element in both ensembles is the Multiple-input Single-output (MISO) predictive LSTM-NN. In both ensembles, the base detectors monitor one specific signal by constantly analyzing the prediction errors received from the predictor, using two distinct techniques. These detectors are the Cumulative Sum Based Detector (CBD) and the Histogram Distance Based Detector (HBD).

The individual decisions of the base detectors are combined using the Adaptive Majority Weighted Voting (AMWV) scheme. A general overview of the proposed ensemble’s architecture is depicted in Figure 2. Here, the building blocks of the ensembles are identified, namely: the predictors, the detectors, and the decision combination method. Each building block will be further described in the following sub-sections.

B. PREDICTORS FEATURE SELECTION

Selecting the appropriate group of signals for the LSTM-NN is an important step towards designing the tampering detection ensembles. The proposed approach follows a correlation-based technique for selecting the groups of inputs and outputs. The selection process is identical for

both proposed detection ensembles, leveraging the Pearson’s product-moment correlation coefficient [30]. Here, from a large number of measured signals, the ones that exhibit a high correlation coefficient with the chosen output signal are selected. Pearson’s product momentum correlation (Pearson’s correlation) describes the strength of the relationship between variables. The process begins with the selection of the output variables (e.g., monitored variables), followed by the selection of the group of variables exhibiting the highest positive and negative correlation coefficients.

Let’s consider X of size n , the set of all measured variables inside a vehicle. Let’s also define Y a subset of X in such a way that $Y \subseteq X$ encompasses the set of output variables. Furthermore, let m be the number of deployed LSTM-NNs. For each j LSTM-NN, let $y^j \in Y$ denote the response variable (e.g., monitored variable), here, y^j can be further selected as input to the other predictors as well. Considering K the number of samples for each variable, given y^j and $x^i \in X$, where $i = 1..n$, the correlation coefficient $R(y^j, x^i)$ is computed as:

$$R(y^j, x^i) = \frac{\sum_{l=1}^K (y_l^j - \bar{y}^j)(x_l^i - \bar{x}^i)}{[\sum_{l=1}^K (y_l^j - \bar{y}^j)^2 \sum_{l=1}^K (x_l^i - \bar{x}^i)^2]^{1/2}}, \quad (1)$$

here, \bar{x}^i and \bar{y}^j denote the average values of x^i and y^j respectively. R can take values between $[-1, 1]$, where the sign of R represents the positive or negative correlation between x^i and y^j . In the following sub-sections, for the LSTM-NN j , where $j = 1..m$, X^j will denote the set of selected input variables.

C. LONG SHORT-TERM MEMORY NEURAL NETWORKS AND TEACHER FORCING

LSTM-NNs are a type of Recurrent Neural Networks (RNN) developed as a solution to the vanishing and exploding gradient problem found in vanilla versions of RNNs. This was achieved by integrating memory units capable of learning when to *forget* and when to *update* memory information. Two core components differentiate LSTM-NNs from other neural networks. First, a sequential input layer capable of feeding data sequences or time-series data to the subsequent layers (e.g., hidden layers). Second, LSTM-NN layers which are responsible for learning dependencies between the time-steps of the data.

A standard LSTM-NN layer is composed of blocks. These blocks incorporate one or more memory cells and three types of gates (e.g., input, output and forget gates). The role of the memory cell is to store information over time and, in term, is controlled by the three gates. These gates regulate the incoming and outgoing information flow to and from the memory cell. The contribution of each gate is as follows: the forget gates, denoted as $f_j(t)$, regulate the discarded information; the input gate, denoted as $l(t)$, regulates what information is to be saved and finally; the output gate, denoted as $o(t)$, computes the current unit’s output.

The equations for the gates, cell update and output, at time t are as follows:

$$f_j(t) = \text{sigm}(W_{f_j}(x(t) + o(t - 1)) + b_{f_j}), \quad (2)$$

$$l(t) = f_2(t) \cdot \bar{C}(t), \quad (3)$$

$$\bar{C}(t) = \text{tanh}(W_C(x(t) + o(t - 1)) + b_{\bar{C}}), \quad (4)$$

$$C(t) = f_1(t) \cdot C(t - 1) + l(t), \quad (5)$$

$$o(t) = f_3(t) \cdot \text{tanh}(C(t)), \quad (6)$$

where W denotes the weight matrices, x is the input vector, \bar{C} are the new candidates for the cell state and C is the current cell state. The activation functions are: the Sigmoid, denoted as sigm and the Hyperbolic Tangent, denoted as tanh and are computed as follows:

$$\text{sigm}(x) = \frac{1}{1 + e^x}, \quad (7)$$

$$\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (8)$$

Training RNNs with Teacher Forcing (TF) involves feeding the ground truth output from the previous time-step $y(t - 1)$, as input to the current time-step, while during testing(inference) instead of the ground truth, the network output $\hat{y}(t - 1)$ is fed back as input. This procedure (with the network outputs fed back as input) has, however, some disadvantages. For instance, the kind of inputs that the network sees during training could be quite different from the kind of inputs that it sees at inference time, this effect is known as exposure bias. The way TF was originally proposed assumes that during inference the ground truth value will not be available. In this paper, a modified version of TF is proposed, where at each time-step, during both training and detection, the ground truth $y(t - 1)$ is fed back as input to the current time-step. During detection, a given signal is monitored by predicting its future values and analyzing the deviations from the ground truth value. Figure 3 illustrates a side-by-side depiction of a generic LSTM-NN block and a LSTM-NN block with TF.

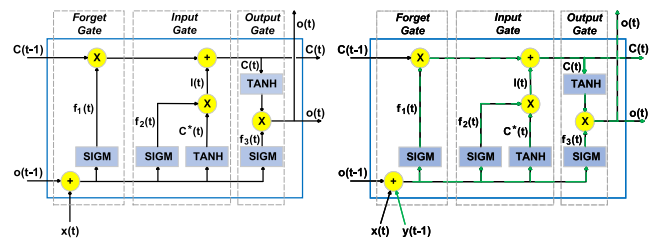


FIGURE 3. LSTM-NN generic block (left) alongside LSTM-NN block with TF (right).

TF is applicable to models that have a recurrent connection from their output leading back into the model and can be used as an alternative to Back Propagation Through Time (BPTT) when the model lacks hidden-to-hidden connections. Nonetheless, TF may still be applied in conjunction with Back Propagation Through Time (BPTT) for training models with hidden-to-hidden connections [31].

The concept behind TF originates from the maximum likelihood criterion [32]. This implies maximizing the conditional log likelihood of the tuple of predictors and responses $\{x_j, y_j\}$, where $j = 1..s$, s is the number of sequences and W denotes the weights of the neural network, as follows:

$$L = \sum_{j=1}^s \log p(y_j|x_j, W), \quad (9)$$

here, the response probabilities are assessed consecutively while each tuple depends on the previous ones. The probability term from the previous equation can further be expanded as:

$$p(y_j|x_j, W) = \prod_{t=1}^{L_j} p(\hat{y}_j(t)|y_j(< t), x_j, W). \quad (10)$$

In Equation 10, the probability of the response $\hat{y}_j(t)$ at time-step t is dependent on both the previous responses $y_j(< t) = \{y_j(0), y_j(1), \dots, y_j(t-1)\}$ and the input sequences x_j .

In our proposed method, the LSTM-NN are trained with both TF and BPTT. The only change in BPTT, using the current version of TF, appears in the forward propagation step as follows:

In Equations 2 and 4 the weight matrices W_{f_i} and W_C are in term composed of the weight matrices for the terms inside the brackets, that is, the current inputs $x(t)$ and the previous state $o(t-1)$. By extending the brackets we get the following:

$$f_j(t) = \text{sigm}(W_{f_j}^x x(t) + W_{f_j}^o o(t-1) + b_{f_j}), \quad (11)$$

$$\bar{C}(t) = \text{tanh}(W_C^x x(t) + W_C^o o(t-1) + b_{\bar{C}}), \quad (12)$$

Let $y(t-1)$ denote the ground truth value at time $t-1$, which will be fed as input at the current time-step t . Then, the weight matrices for the inputs can be separated in two terms: the weight matrix for the current external inputs x and the weight matrix for the previous ground truth value $y(t-1)$. Equations 11 and 12 are rewritten as:

$$f_j(t) = \text{sigm}(W_{f_j}^y y(t-1) + W_{f_j}^x x(t) + W_{f_j}^o o(t-1) + b_{f_j}), \quad (13)$$

$$\bar{C}(t) = \text{tanh}(W_C^y y(t-1) + W_C^x x(t) + W_C^o o(t-1) + b_{\bar{C}}), \quad (14)$$

the same changes will also apply to Equations 3, 4 and 6. The current hidden state $o(t)$, which is also the current output $\hat{y}(t)$, becomes:

$$\begin{aligned} o(t) &= f_3(t) \cdot \text{tanh}(C(t)) \\ &= \text{sigm}(W_{f_3}^y y(t-1) + W_{f_3}^x x(t) \\ &\quad + W_{f_3}^o o(t-1) + b_{f_3}) \cdot \text{tanh}(C(t)) \end{aligned} \quad (15)$$

It becomes obvious, from Equation 15, that the current hidden state, which is also the output, quantifies all previous hidden states together with the previous ground truth values, and this applies to both the training and detection phases.

As there are no recurrent connections and weights from the output layer to the input layer, the backwards propagation is not affected. For a more detailed description of LSTM-NN and the BPTT algorithm the reader is encouraged to examine [33], [34], [35].

In this paper, the predictor is defined as the LSTM predictive neural network with an additional component that computes and outputs the prediction error at each time-step, further defined as $\epsilon = y(t) - \hat{y}(t)$.

D. BASE DETECTORS ARCHITECTURE

The current subsection will describe the architecture of the base detectors, namely, the Cumulative Sum Based Detector (CBD) and the Histogram Distance Based Detector (HBD).

1) CUMULATIVE SUM BASED DETECTOR

The CBD monitors changes, in both the mean and the variance values of the LSTM-NN prediction error, using two variants of the 1-CUSUM scheme [36]. The 1-CUSUM scheme has the ability to detect changes (i.e., increase and decrease shift) in both the mean and the variance values, using a single two-sided control chart, which works with single observations. The first proposed variant utilizes the CUSUM chart as it was originally proposed in [36], for a point-by-point CUSUM computation, while the second variant employs a sliding window methodology, which computes the CUSUM over a sliding window of size τ .

Let μ_0 be the mean value of the training prediction error ϵ , and let $\nu = \epsilon - \mu_0$. The CUSUM value, further denoted as $CSM(t)$, at time t , is computed as follows:

$$\begin{aligned} CSM(t) &= \max[0, CSM(t-1) + (\lambda\nu(t) + (1-\lambda)\nu^2(t)) - \beta], \\ &\quad \text{if } (CSM(t-1) > 0 \text{ or} \\ &\quad (CSM(t-1) = 0 \text{ and } \nu(t) > 0)) \end{aligned}$$

or,

$$\begin{aligned} CSM(t) &= \min[0, CSM(t-1) + (\lambda\nu(t) - (1-\lambda)\nu^2(t)) + \beta], \\ &\quad \text{if } (CSM(t-1) < 0 \text{ or} \\ &\quad (CSM(t-1) = 0 \text{ and } \nu(t) < 0)). \end{aligned} \quad (16)$$

In Equation 16, β is the reference parameter value, λ ($0 \leq \lambda \leq 1$) is the weighting factor and $CSM(0)$ is initialized with 0. The algorithm for computing the CUSUM over a sliding window is presented in Algorithm 1.

The detection threshold for each detector, denoted as θ_j , is computed as $\theta_j = \mu_{CSM_j} + 6 \cdot \sigma_{CSM_j}$. Here, μ_{CSM_j} and σ_{CSM_j} , denote the mean and standard deviation of the cumulative sum computed over the training data prediction errors.

During the detection phase, the CBD constantly computes the CUSUM value, over the prediction error received from the predictor, and outputs a binary decision which can be: clean,

Algorithm 1: Sliding Window Cumulative Sum

Data:
 ϵ : Prediction absolute errors;
 μ_0 : Training prediction error mean value;
 τ : Sliding window size;
 β : Reference parameter value;
 λ : Weighting factor;

Result:
 CSM : The cumulative sum;

Function CUSUM ($\epsilon, \mu_0, \tau, \beta, \lambda$):
For $i \leftarrow 1$ **to** $\text{size}(\epsilon) - \tau$ **do**
 $idx \leftarrow 1$;
 $WC[idx] \leftarrow 0$; // CUSUM over the sliding window.
 For $j \leftarrow i$ **to** $i + \tau - 1$
 $idx \leftarrow idx + 1$;
 $v \leftarrow \epsilon[j] - \mu_0$;
 if $WC[idx - 1] > 0$ **or** $(WC[idx - 1] == 0 \text{ and } v > 0)$
 $WC[idx] \leftarrow \max\{0, WC[idx - 1] + (\lambda v + (1 - \lambda)v^2) - \beta\}$;
 else if $WC[idx - 1] < 0$ **or** $(WC[idx - 1] == 0 \text{ and } v < 0)$
 $WC[idx] \leftarrow \min\{0, WC[idx - 1] + (\lambda v - (1 - \lambda)v^2) + \beta\}$;
 End
 End
 $CSM[i] \leftarrow WC[idx]$;
End
Return CSM
End Function

if the computed CUSUM remains under the threshold value, or tampered, if it surpasses it.

2) HISTOGRAM DISTANCE BASED DETECTOR

The base detector of the second ensemble, named HBD, also utilize LSTM-NN predictive models but processes the prediction errors differently. Firstly, the HBD constructs the histogram of the prediction errors, over a given time window. Secondly, using a custom distance metric it computes the distance between the histogram of the prediction errors of the training data and the histogram of the data contained in the current time window. Lastly, each detector outputs a binary decision using a threshold-based approach.

The proposed histogram distance metric computes the maximum absolute distance between two given histograms. Let H and U be two histograms, defined as $H = (h_1, h_2, \dots, h_\psi)$ and $P = (p_1, p_2, \dots, p_\psi)$, having the same number of bins, denoted as ψ , with the same bin edges BE , where, h_i and p_i represent the frequencies of points in bin i . The distance between the two histograms, further denoted as $d(H, P)$, is computed as follows.

$$d(H, P) = \max_{h_i \in H, p_i \in P} \{|h_i - p_i|\}. \quad (17)$$

Before applying the proposed distance metric, let us first prove that it is a metric.

A distance $d(H, P)$ defined in a space of dimension \mathbb{R}^n , $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ must satisfy the following properties, in order to be considered a metric:

Property 1 (Non-negativity): $d(H, P) \geq 0$.

Proof: The distance $d(H, P)$ is the maximum of the absolute differences between the probability of points in each bin. The absolute value of each $|h_i - p_i|$, where $h_i \in H$ and $p_i \in P$ has a non-negativity property by the definition of the

absolute value, where:

$$|a| = \begin{cases} a & \text{if } a \geq 0 \\ -a & \text{if } a < 0, \end{cases} \quad (18)$$

Therefore, $d(H, P) = \max_{h_i \in H, p_i \in P} \{|h_i - p_i|\}$

also has the non-negativity property by definition. \square

Property 2 (Reflexivity): $d(H, H) = 0$.

Proof: Since $H = H$ then each value $h_i = h_i$, where $h_i \in H$. If each $h_i = h_i$ then each $h_i - h_i = 0$ and each $|h_i - h_i| = 0$.

Therefore, $d(H, H) = \max_{h_i \in H} \{|h_i - h_i|\} = 0$ by definition. \square

Property 3 (Commutativity): $d(H, P) = d(P, H)$.

Proof: Given the symmetry propriety of the absolute value, where, $|a| = |-a|$ by substituting $(H - P)$ for a , we can write $|(H - P)| = |-(H - P)|$. This means that for each $h_i \in H$ and $p_i \in P$ we can write $|(h_i - p_i)| = |-(h_i - p_i)| \Rightarrow |(h_i - p_i)| = |(p_i - h_i)|$. Since $|(h_i - p_i)| = |(p_i - h_i)|$ then

$$\max_{h_i \in H, p_i \in P} \{|h_i - p_i|\} = \max_{h_i \in H, p_i \in P} \{|p_i - h_i|\}.$$

Therefore, $d(H, P) = d(P, H)$. \square

Property 4 (Triangle Inequality): $d(H, Q) \leq d(H, P) + d(P, Q)$.

Proof:

Let's begin by fixing $i = 1, \dots, n$.

Then, we have $|h_i - q_i| = |h_i - p_i + p_i - q_i| \leq |h_i - p_i| + |p_i - q_i|$ by definition of the Triangle Inequality in \mathbb{R} .

Next, we take the maximum of both sides of the inequality which gives us:

$$\begin{aligned} \max_{h_i \in H, q_i \in Q} \{|h_i - q_i|\} &= \max_{h_i \in H, p_i \in P, q_i \in Q} \{|h_i - p_i + p_i - q_i|\} \\ &\leq \max_{h_i \in H, p_i \in P} \{|h_i - p_i|\} + \max_{p_i \in P, q_i \in Q} \{|p_i - q_i|\}. \end{aligned}$$

Therefore, $d(H, Q) \leq d(H, P) + d(P, Q)$. \square

The algorithm for HBD with a sliding window approach is presented in Algorithm 2. In the same Algorithm, $H(\epsilon_0)$ denotes the histogram of the training prediction errors, $H(\epsilon_0) = (h_1, h_2, \dots, h_\psi)$, here, h_j where $j = 1.. \psi$, represents the frequencies of points in bin j , and ψ denotes the total number of bins. Also, $H(\epsilon_\tau)$ denotes the histogram of the data points from the current time-window.

The threshold computation for the HBD begins with the construction of the histogram distance vector, over the training prediction errors, using the approach presented in Algorithm 2. The threshold value is computed as: $\theta_j = \mu_z \pm 6 \cdot \sigma_z$, here, μ_z and σ_z denote the mean and the standard deviation of the histogram distance vector z . The same procedure is followed for all the detectors in the ensemble.

During the detection phase, the detectors will constantly compute the distance between the training histogram and the histogram of the prediction error from the current time window. Similarly to the CUSUM based approach, the detectors will output a binary decision, namely, clean or tampered,

Algorithm 2: Sliding Window Histogram Distance

Data:
 ϵ : Prediction absolute errors;
 τ : Sliding window size;
 ψ : Histogram number of bins;
 $H(\epsilon_0)$: Histogram of the training prediction errors;
 BE : Bin edges from $H(\epsilon_0)$;
Result:
 z : Histogram distance vector;
Function HistDist ($\epsilon, \tau, \psi, H(\epsilon_0)$):
For $i \leftarrow 1$ **to** $\text{size}(\epsilon) - \tau$ **do**
 $H(\epsilon_\tau) = \text{Histogram}(\epsilon[i : i + \tau - 1], \psi, BE)$;
 $z[i] = d(H(\epsilon_0), H(\epsilon_\tau))$;
End
Return z
End Function

based on the computed histogram distance and the detection threshold.

E. ADAPTIVE MAJORITY WEIGHTED VOTING

The proposed fusion technique is a modified version of the Majority Weighted Voting scheme, with an additional historic reputation component. That is, at each time-step the weights of the detectors are adjusted (e.g., increased or decreased) depending on whether the detector votes the same as the majority or not. The weights are adjusted by a percentage, which is computed based on the previous decisions of said detector (e.g., if in the past, it has voted the same as the majority). This weight adjustment methodology awards higher weights to the detectors that vote the same as the majority compared to the detectors that more often disagree with the majority. Furthermore, the weights can drop down to zero, thus temporarily *ignoring* that detectors decision.

The final decision of the ensembles can be one of the following: normal, alert and warning. The warning state is triggered when the majority vote doesn't trigger an alert but there is at least one detector that identified a tampered observation. Considering that each detector monitors a different signal, there is the possibility that tampering one or more signals might not affect the rest of signals, especially if the physical tampered component is not running in a closed-loop. Thus, by generating a warning, further investigations can be carried out on that component.

Recall, each detector of the ensembles outputs a binary decision, namely, clean or tampered. Let k denote the total number of detectors of an ensemble and let \mathcal{D}_j , where $j = 1..k$, be the j -th detector. The output of detector \mathcal{D}_j is denoted as $\Omega_{j,i} \in \{0, 1\}$, here, $i \in \{1, 2\}$ represents the two possible decisions (e.g., 1: clean, 2: tampered). $\Omega_{j,i}$ will take the value of 0, if detector j outputs a clean decision, and 1 otherwise. These decisions are further passed on to the voting system which sums the weights, for each decision, and outputs the

decision that receives the most weighted votes, as follows:

$$V(t) = \begin{cases} 1, & \text{if } \sum_{j=1}^k w_j(t)\Omega_{j,1} > \sum_{j=1}^k w_j(t)\Omega_{j,2}, \\ 2, & \text{otherwise,} \end{cases} \quad (19)$$

where $V(t)$ denotes the majority weighted decision at time t and w_j denotes the current weight of detector \mathcal{D}_j . The final ensemble output at time t , denoted as $V_E(t)$, is computed as follows:

$$V_E(t) = \begin{cases} \text{Normal,} & \text{if } V(t) = 1, \\ \text{Warning,} & \text{if } V(t) = 1 \text{ and } \sum_{j=1}^k \Omega_{j,2} > 0, \\ \text{Alert,} & \text{if } V(t) = 2. \end{cases} \quad (20)$$

In case of a draw between Normal and Alert, the decision will be: Alert (tampered).

The weights for the next time-step, $w_j(t + 1)$, are adjusted as follows:

$$w_j(t+1) = \begin{cases} \min \left(1, w_j(t) + w_j(t) \frac{1}{1 + \sum_{l=1}^p [A_{j,l} == 0]} \right), & \text{if } \mathcal{D}_j \text{ voted the same as the majority.} \\ \max \left(0, w_j(t) - w_j(t) \frac{1}{1 + \sum_{l=1}^p [A_{j,l} == 1]} \right), & \text{otherwise,} \end{cases} \quad (21)$$

here, A is a matrix of size $k \times p$ denoting the historical decisions for the detectors, namely, each row represents a detector, and each column holds a value of 1 if the detector voted the same as the majority and 0 otherwise, in the previous p time-steps.

In Equation 21, the first case represents the scenario where a detector's decision is the same as the majority, so its weight is increased. In the denominator $\sum_{j=1}^p [A_{i,j} == 0]$ represents the number of times the detector disagreed with the majority in the past p time-steps. Conversely, the second case represents the scenario where a detector disagrees with the majority and $\sum_{j=1}^p [A_{i,j} == 1]$ is the number of times the detector agreed with the majority in the past p time-steps. If a detector's weight drops to zero and after a while it starts voting the same as the majority, its weight is initialized to a value of 0.1.

IV. MATERIALS AND METHODS

A. DATASETS AND TAMPERING SCENARIOS

As previously mentioned, the proposed ensembles were tested and validated on three independent datasets. Each dataset originates from different environments (e.g., vehicle or simulation), containing distinct tampering scenarios.

The primary dataset was used for evaluation and validation, while the two secondary datasets were used for evaluation, validation, and performance comparison with state-of-the-art tampering detection techniques.

1) PRIMARY DATASET

The first dataset was produced by a state-of-the-art aftertreatment simulation model of a heavy-duty vehicle developed at the Laboratory of Applied Thermodynamics from Aristotle University of Thessaloniki.

Here, a simulation was developed, to produce vehicle data covering multiple driving cycles and tampering scenarios. Using the Exothermia suite simulation environment [37] a bus incorporating a state-of-the-art exhaust aftertreatment system was created. This bus incorporates the major components found in heavy-duty aftertreatment systems: an inline Diesel Oxidation Catalyst (DOC), Selective Catalytic Reduction (SCR) Filter, SCR system, and Ammonia Slip Catalyst (ASC). Exothermia suite has the ability to model the flow through the systems of emission control systems using physicochemical solvers. Among the inputs of the model we find the mission profile (e.g., target speed, road slope and diverse environmental conditions such as temperature, pressure and humidity). The simulation outputs several parameters, including emissions, exhaust gas temperature or urea dosage [38].

In the above-mentioned system, two driving cycles were created: the World Harmonized Vehicle Cycle (WHVC), and the Fige transient cycle. The WHVC represents a heavy-duty chassis dynamometer test developed mainly for research purposes and utilized for engine emissions certifications. It comprises 1800 seconds of measurements, divided into three segments which include urban, rural and motorway driving [39]. Developed by the FIGE Institute, from Aachen, Germany, the Fige transient cycle is a truck and bus engine test cycle, based on real measurements of heavy-duty vehicles [40]. Alike the WHVC cycle, Fige cycle comprises 1800 seconds which include urban, rural and motorway driving segments. All simulations were carried out at a 10Hz sampling frequency for all signals.

To create tamper-free (baseline) evaluation scenarios, the two driving cycles were simulated without modifications. Next, the tampering scenarios were created. They include known and unknown (possible future) tampering approaches and follow two main operating modes. The first operating mode refers to disabling EAS components and hiding the affected signals by injecting emulated signals resembling the real ones. The second operating mode considers more advanced emulators that could function in training mode as well, where the emulator attempts to model and fit the emulated signals to the real ones.

A significant tampering scenario, frequently observed in heavy-duty vehicles, involves disabling the NOx reduction system for a decrease in maintenance costs (e.g., refiling the AdBlue DEF). This specific tampering scenario was simulated by recreating an AdBlue emulator, which was

previously tested on a Renault MDA2C EuroVI truck, while known and possible future hiding methods were applied to the affected signals. This emulator performs two actions, namely, it reduces the Urea dosing command while concurrently creating two hiding signals targeting the Engine Control Module. Here, the hiding methods range from simpler approaches like injection of NOx downstream values as a fixed and random percentage of the NOx upstream values, to more complex ones, like fitting the emulated signals using multiple linear regression and moving average techniques. To stay undetected from plausibility and tampering detectors, these advanced models provide emulated signals with patterns resembling the ones of functional exhaust aftertreatment system. Nonetheless, as the complexity of such models increases so does the computational demand together with the need for additional available signals for an efficient training. The resulting tampering scenarios, further denoted as T1 - T6, include 6 hiding methods applied to the NOx downstream signal combined with one tampering method applied to the Urea signal. The tampering scenarios fit the emulated signals as follows, T1 - T2 using a fixed and random percentages of the upstream values, T3 - T4 using moving average techniques and T5 - T6 using linear regression. A more detailed description of the simulation model and the tampering scenarios can be found in [38].

2) SECONDARY DATASET I

The second dataset was used for two reasons, first, to evaluate the proposed ensembles on a dataset that originates from a real vehicle, and second, for the performance comparison experiments. This dataset originates from the original VetaDetect paper [23] and it comprises data collected from a EURO VI D N2 class truck in the Vehicle Emissions Heavy Duty chassis laboratory (VELA) at the Joint Research Centre of the European Commission. This dataset contains emission-related tamper-free and tampered data measured while using an AdBlue emulator mounted inside the vehicle. Both baseline and tampered scenarios were tested using the WHVC driving cycle and contain 1800 seconds of measurements. Furthermore, the authors also included a more advanced tampering scenario, where the tamperer uses an ARX model to emulate and replace the readings of an Intake Oxygen sensor. Additionally, another scenario was included, where the tamperer might use a more complex model (e.g., an LSTM network trained with tamper-free measurements from the same truck) to emulate the Intake Oxygen sensor.

3) SECONDARY DATASET II

Moving forward to the third dataset, created by Roman et al. [24]. Here, the authors collected their data from the On-Board Diagnostic II (OBD-II) port of a 2015 EUR6 Skoda Rapid 1.2 L TSI passenger vehicle. They used an OBD-II USB interface cable together with an aftermarket diagnostic software (e.g., VCDS) to collect the data. The tamper-free collection contains measurements from 12 in-vehicle sensors, while the tampered collections include replay

tampering methods on multiple signals, such as: oxygen, coolant temperature, engine torque and throttle value position. Furthermore, the authors included several tampering scenarios, ranging from tampering one sensor all the way to four concurrent sensors. Additionally, each of the mentioned collections having two versions: original and anonymized. This dataset was also used for the performance comparison experiments.

B. DATA PRE-PROCESSING

As a first step, the constant and duplicate features were removed from the datasets, this was followed by data normalization. As the range of the features varies significantly, data normalization is needed to bring all the features to the same scale. Each feature was normalized using the feature scale method (e.g., bringing all values in [0, 1] range), as shown in Equation 22.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (22)$$

here, x' is the normalized value for a given feature x and x_{min} , x_{max} denote the minimum and maximum values of x . Furthermore, the scaling is done independently for each feature.

The feature scaling method, as described above, is however applicable only for the training dataset or for offline testing, as it uses the dataset's minimum and maximum values for each feature. In a real-time scenario though, the incoming data needs to be normalized without having access to the whole dataset. To address this issue, the minimum and maximum vectors from the training dataset are stored and used for scaling the rest of the datasets. For real-world usage, considering the fact that different systems work on different ranges, the maximum and minimum values would be taken from the specifications of each vehicle model.

1) SELECTING THE OPTIMUM NUMBER OF BINS

Selecting the optimum number of bins ψ , as it is showed in this section, is important for the performance of the Histogram Based Ensemble (HBE). A small number of bins, each covering a larger range of values, might cause an incorrect grouping of clean and tampered points in the same bin, producing higher false negative detections. Conversely, a large number of bins, each covering a smaller range of values, might cause clean points to be inserted in empty bins, producing false positive detections.

In contrast, the number of observations (e.g., window size) also influences the detection procedure. Recall Figure 1, the tampered observations closely resemble the tamper-free ones, using a small number of points might not fit the distribution correctly and the changes in the data distribution will not be captured.

Another aspect that needs consideration is that the threshold computation methodology involves constructing the histogram of the prediction error over the entire training set, which naturally is larger than the window size used during evaluation. Thus, it is important to observe how the

relationship between the window size and the number of bins, affects the performance.

In order to observe the performance variations, several well known bin selection rules were identified, so called *rules of thumb*. Authors who have suggested selection rules include Freedman and Diaconis [41], Terrell et al. [42], Scott [43] and Sturges [44]. These techniques are quite popular, most of them being integrated in well known numerical and statistical frameworks (e.g., by default the R framework uses Struge's rule, while MATLAB offers out-of-the-box implementation for all of the above mentioned rules). Other papers, which proposed histogram based outlier score techniques [45], [46], [47], suggest using the square root method, namely, setting the number of bins to be equal to the square root of the number of samples. Others have also proposed using powers of two as the number of bins in their experimental assessment [48], [49]. The above presented bin selection methods are summarized in Table 1.

TABLE 1. Selection methods, for the number of histogram bins, found in the literature. In this table, N denotes the number of observations, Y denotes the observation vector, σ_Y denotes the standard deviation of Y and ψ denotes the number of bins computed for 15000 observations.

Method	Equation	Computed ψ
Diacomis-Freedman [41]	$\left\lceil 2 \cdot \frac{IQR(Y)}{\sqrt[3]{N}} \right\rceil$	123
Terrell-Scott [42]	$\lceil \sqrt[3]{2 \cdot N} \rceil$	32
Rice [50]	$\lceil 2 \sqrt[3]{N} \rceil$	50
Scott [43]	$\left\lceil \frac{3.5 \cdot \sigma_Y}{\sqrt[3]{N}} \right\rceil$	307
Sturge [44]	$\lceil 1 + \log_2 N \rceil$	15
Square Root	$\lceil \sqrt{N} \rceil$	123

When selecting the number of bins ψ , the following procedure was considered: a base detector and a predictor were selected, together with a tamper-free test dataset, this was followed by measuring the False Positive Rate of the detector using different values for ψ while modifying the number of observations (e.g., window size). Figure 4 depicts the results in terms of False Positive Rate with ψ ranging from 2 to 512 (with highlighted colors showing the number of bins, computed using various well known methods for the largest window size of 15000) and the number of observations ranging from 100 to 15000. As depicted in Figure 4, in this specific scenario, for 32 bins (Terrel-Scott Rule) and for 15 bins (Sturge's Rule) the False Positive Rate on average remains close to zero for all window sizes (number of points), thus providing two feasible number of bin selection methods.

C. PROPOSED ENSEMBLES DESIGN

This section describes the final architectures of the two proposed ensembles, including the common components, such as, the configuration of the predictors and the hyper-parameters, as well as ensemble specific parameters, the threshold values, window sizes and histogram specific

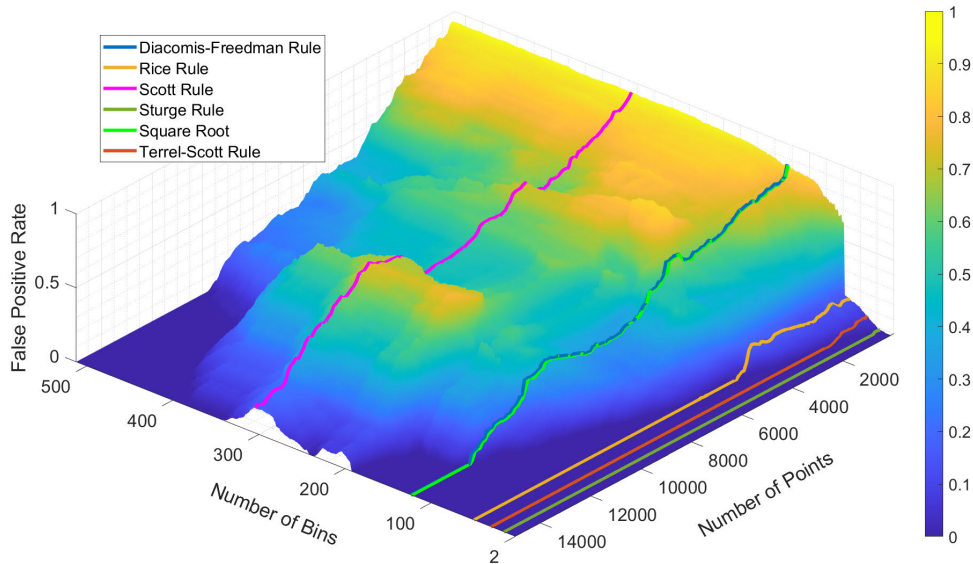


FIGURE 4. False Positive Rate vs Number of Bins vs Number of Points.

TABLE 2. The complete list of selected hyper-parameters for the LSTM-NN.

Predictor	Hidden Layers		Hidden Units		Learning Rate		Epochs	
	Range	Final value	Range	Final value	Range	Final value	Range	Final value
1	[1, 3]	1	[1, 100]	16	[0.0001, 0.1]	0.01	[0, 500]	50
2	[1, 3]	1	[1, 100]	16	[0.0001, 0.1]	0.015	[0, 500]	40
3	[1, 3]	1	[1, 100]	16	[0.0001, 0.1]	0.015	[0, 500]	40
4	[1, 3]	1	[1, 100]	16	[0.0001, 0.1]	0.016	[0, 500]	50

parameters (e.g., the histogram length and number of bins). These architectures were used for the evaluation on the primary dataset, the architecture of the predictors used on the secondary datasets is presented in Subsection V-C.

Both ensembles consist of 4 base detectors, monitoring the NO_x Outlet ASC, Oxygen ASC, Oxygen SCR and the Urea dosage command signals by processing the prediction residuals. These four signals represent the responses (outputs) of 4 predictors. The complete list of selected inputs and outputs, for the predictors, is summarized in Table 4. For LSTM hyper-parameter tuning, two techniques were selected, namely, Bayesian optimization [51] and parameter sweeping (e.g., grid search). The final values for the hyper-parameters and the search ranges used during tuning are presented in Table 2. Apart from these, the Adam optimizer was used [52], and the following parameters were manually selected: Batch size: 128, Learning Rate drop period: 10 epochs, Learning Rate drop factor: 10%.

For the Primary Dataset, the LSTM-NNs were trained using only tamper-free observations originating from the WHVC dataset using a 90:10 split, namely, 90% of the data was used for training and 10% for validation during training, while the ensembles were tested on the tamper-free and tampered variants of the WHVC and Fige datasets.

In total, 14 datasets were used, namely, the 2 tamper-free and 6 tampered variants for each of the WHVC and Fige driving cycles. The tampered datasets are as follows: each driving cycle consisted of 6 tampered datasets (T1 - T6) containing 6 different NO_x hiding methods combined with 1 Urea hiding method, as described in Section IV-A1.

For the Secondary Datasets I, as this dataset contains only one tamper-free collection, the LSTM neural networks were trained using 60% of the tamper-free collection. During testing, the remaining 40% of the tamper-free collection and the tampered variants of the dataset were used.

Finally, for the Secondary Dataset II, similarly to the previous case, the LSTM neural networks were trained using 60% of the tamper-free collection while during testing the remaining 40% of the tamper-free collection, the entire anonymized versions of the dataset and the tampered variants of the dataset were used.

Another two common parameters between both ensembles are the length of the historical vector and the initial weights used by the fusion system, in all experiments the value for the former was 10 while for the latter it was 1. The remaining parameters are ensemble specific and are summarized in Table 3.

TABLE 3. The final values for the CBD and HBD parameters.

Parameter	Description	CBD	CBD with Sliding Window	HBD
τ	Sliding Window Size	-	4000	12000
λ	CUSUM Weighting Factor	0.8	0.8	-
β	CUSUM Reference	0.5	0.5	-
ψ	Number of Bins	-	-	32

V. EXPERIMENTS AND EVALUATION

The experimental assessment consists of two distinct environments:

- (i) Prototype environment.
- (ii) Embedded environment.

In the Prototype environment (i), the ensembles were implemented in MATLAB R2022a, running on a Lenovo Legion laptop with an AMD Ryzen 5 5600H CPU, 16 GB RAM DDR4 running Windows 10 PRO. Here, the solutions were evaluated on both the tamper-free and tampered datasets using the metrics described in Section V-A.

As for the Embedded environment (ii), the ensembles were implemented in python using TensorFlow and Keras, on a Raspberry Pi 4 model B board, having 8 GB RAM and running Raspbian OS. In this case, the aim was to measure the resource consumption in a real embedded environment. These measurements include CPU and RAM usage, only during the detection phase, since the training is done offline. Offline training would reduce the resource requirements whilst running the ensemble in a real automotive environment. To measure the CPU and RAM usage, *psutil* [53] was used. *Psutil* is a cross-platform library for Python, used for retrieving information on running processes and system resource utilization (e.g., CPU, memory, disks, network, sensors).

A. PERFORMANCE EVALUATION METRICS

For a complete in-depth performance evaluation of the proposed detection methodologies, the following performance metrics are used:

- **Accuracy.** Accuracy is defined as the ratio between the correctly classified observations to the total observations.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \quad (23)$$

- **Specificity (True Negative Rate).** Specificity is defined as the number of correctly classified negative observations out of all the true negative observations.

$$TNR = \frac{TN}{TN + FP}. \quad (24)$$

- **Recall (True Positive Rate).** Recall is defined as the number of correctly classified positive observations out of all the true positive observations.

$$TPR = \frac{TP}{TP + FN}. \quad (25)$$

- **False Positive Rate.** False Positive Rate is the proportion of negative observations incorrectly identified as positive observations (i.e. the probability of false alerts will).

$$FPR = \frac{FP}{FP + TN} = 1 - TNR. \quad (26)$$

- **False Negative Rate.** False Negative Rate is the proportion of positive observations incorrectly identified as negatives observations.

$$FNR = \frac{FN}{FN + TP}. \quad (27)$$

- **Precision (Positive Predicted Value).** Precision is defined as the fraction of correctly classified positive observations of all predicted positive observations.

$$PRC = \frac{TP}{TP + FP}. \quad (28)$$

- **F1 score (F-Measure).** F1-Score is defined as the harmonic mean of precision and recall. It is a statistical measure of the accuracy of a test or a model.

$$F1 = \frac{2 * PRC * TPR}{PRC + TPR}. \quad (29)$$

- **AUC-ROC.** The area under the receiver operating characteristic (ROC) curve, denoted as AUC, is an performance evaluation metric defined as the integral of a ROC curve (i.e., TPR) with respect to the FPR from FPR = 0 to FPR = 1.

The above performance metrics are dependent on the following: true positives (TP), denoting the number of correctly detected positive observations; the true negative (TN), the number of negative values that are detected as negatives; false negative (FN), the number of negative values that are falsely detected; and, false positive (FP), the number of negative values that are categorized as positives. In this paper the positive class denotes the tampered (anomalous) observations while the negative class denotes the clean (tamper-free) observations.

B. DETECTION RESULTS

The current section presents the detection results of the proposed ensembles on the primary dataset, as described in Section IV-A. Recall, this dataset contains 14 collections, out of which 2 contain tamper-free observation, 1 for each

TABLE 4. The complete list of selected inputs and outputs, for the LSTM predictive neural networks, on the Primary Dataset.

Predictor	Output	Input Variables
1	NOx Outlet ASC	Engine Speed Lambda NH ₃ Outlet ASC NOx Conversion SCRF NOx Intake SCR Urea Command
2	O ₂ Intake ASC	Driver AT Accelerator Pedal Engine Torque Lambda NOx Intake DOC NOx Intake SCRF NOx Outlet ASC Temperature Intake DOC Temperature Intake SCRF Urea Command Vehicle Speed
3	O ₂ Intake SCR	Driver AT Accelerator Pedal Engine Torque Lambda NOx Intake DOC NOx Intake SCRF NOx Outlet ASC Temperature Intake DOC Temperature Intake SCRF Urea Command Vehicle Speed
4	Urea command	Driver AT accelerator pedal Engine Torque Lambda NH ₃ Outlet ASC O ₂ Intake DOC O ₂ Intake ASC Pressure Intake DOC Temperature Intake DOC Temperature Intake ASC Vehicle Speed

driving cycle (e.g., WHVC and Fige), while the remaining 12 collections contain 6 types of tampering on the NOx signal combined with 1 type of tampering on the Urea command signal, for each driving cycle.

The results for the first variant of CBE, using CUSUM based point-by-point detectors, using 8 performance metrics, is summarized in Table 5. In the absence of tampering, on the tamper-free scenarios, the CBE obtained 0% FPR and 100% TNR, on both driving cycles. Concurrently, in the presence of tampering, the detection rate was high on all tampering scenarios, for both driving cycles (>99.75%), with a Precision score of 100% for all the experiments and over 99.8% for the F1 Score, Accuracy and AUC.

Moving forward to the evaluation of the second variant of the CBE, using a sliding window approach. In the absence of tampering, similarly to the first CUSUM based variant of the ensemble, the FPR was 0% with an 100% TNR on both driving cycles, while in the presence of tampering, the CBE with sliding window performed better than the first variant, obtaining 100% for all other metrics. Compared to the point-by-point variant, the sliding window approach, demonstrated

a maximum increase of 0.24% in terms of TPR and a maximum increase of 0.12% in terms of Accuracy on the WHVC driving cycle. Simultaneously, on the Fige driving cycle, the sliding window CBE exhibits a 0.14% increase in terms of TPR and a 0.07% increase in terms of Accuracy and F1 score. Table 6 summarizes the evaluation results for the sliding window CBE, on both driving cycles.

Lastly, the evaluation results for the HBE are summarized in Table 7. In a similar manner to the other variants of the ensemble, in the absence of tampering, the HBE obtained a 0% FPR on both driving cycles, together with a 100% TNR. In the presence of tampering, HBE obtained similar results to the sliding window CBE, scoring 100% on all the metrics, on all tampering scenarios. Compared to the the point-by-point CBE, an increase of 0.24% in terms of TPR and 0.12% in terms of Accuracy, is observed on the WHVC driving cycle. On the Fige driving cycle, the HBE evaluation reveals an increase of 0.14% in terms of TPR and 0.07% in terms of Accuracy, F1 score and AUC.

C. COMPARISON WITH STATE-OF-THE-ART SOLUTIONS

Both proposed detection ensembles were compared to state-of-the-art detection techniques, namely, to VetaDetect [23] and to the detection technique in [24]. The evaluation methodology focused on measuring and comparing both the False Positive Rate and the True Positive Rate on the original datasets from the two papers, these datasets were detailed in Section IV-A (Secondary Dataset I and Secondary Dataset II). The motivation behind choosing these two metrics (e.g., FPR and TPR) lies in the fact that the authors of both papers measured the performance of their models using only these metrics.

The results for the first comparison, on the Secondary Dataset I, between the proposed ensembles and VetaDetect [23], are summarized in Table 8 and the configuration of the predictors in summarized in Table 9.

On tamper-free dataset, the three flavors of the proposed ensembles, CBE, CBE with sliding window and HBE all yielded 0% FPR, a decrease of 0.38% compared to VetaDetect. Advancing to the first tampering scenario, where a real emulator was used, CBE with sliding window and HBE obtained similar results to VetaDetect, showing 100% detection rate (e.g., TPR), while the CBE, with point-by-point detection obtained a TPR rate of 82.14%, a difference of 17.86% compared to the other ensembles.

Moving forward to the advanced tampering scenarios. Here, the first scenario involves that the tamperer might use ARX trained models to emulate a variable (Intake O₂). In this case, the tamperer disables the actual O₂ sensor and emulates the normal behavior of the sensor, using ARX models. Here, the TPR of the CBE was 99.52%, with an increase of 1.02% compared to VetaDetect, while the CBE with sliding window and HBE obtained a respectable 100% detection rate, an increase of 1.5% compared to VetaDetect and a 0.42% increase compared to the CBE. Finally, in the last advanced tampering scenario, where the tamperer fits and deploys an

TABLE 5. Evaluation results for two driving cycles (WHWC, Fige) on clean and 12 tampering scenarios, using the CBE (CUSUM Based Ensemble) with point-by-point CUSUM computation.

	WHWC								Fige							
	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC
Clean	1	0	-	-	-	-	-	-	1	0	-	-	-	-	-	-
T1	-	-	0.9981	0.0019	0.9990	1	0.9990	0.9990	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993
T2	-	-	0.9981	0.0019	0.9990	1	0.9990	0.9990	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993
T3	-	-	0.9981	0.0019	0.9990	1	0.9990	0.9990	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993
T4	-	-	0.9981	0.0019	0.9990	1	0.9990	0.9990	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993
T5	-	-	0.9976	0.0024	0.9988	1	0.9988	0.9988	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993
T6	-	-	0.9980	0.002	0.9990	1	0.9990	0.9990	-	-	0.9986	0.0014	0.9993	1	0.9993	0.9993

TABLE 6. Evaluation results for two driving cycles (WHWC, Fige) on clean and 12 tampering scenarios, using the CBE (CUSUM Based Ensemble) with a sliding window CUSUM computation.

	WHWC								Fige							
	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC
Clean	1	0	-	-	-	-	-	-	1	0	-	-	-	-	-	-
T1	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T2	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T3	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T4	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T5	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T6	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1

TABLE 7. Evaluation results for two driving cycles (WHWC, Fige) on clean and 12 tampering scenarios, using the HBE (Histogram Based Ensemble).

	WHWC								Fige							
	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC	TNR	FPR	TPR	FNR	ACC	PRC	F1	AUC
Clean	1	0	-	-	-	-	-	-	1	0	-	-	-	-	-	-
T1	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T2	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T3	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T4	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T5	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1
T6	-	-	1	1	1	1	1	1	-	-	1	1	1	1	1	1

LSTM based predictor to emulate the Intake O_2 . Similarly to the first advanced tampering scenario, the tamperer here disables the O_2 sensor and emulates the normal behavior of the sensor, using LSTM models. In this scenario, VetaDetect obtained a TPR of 64.46% while the proposed ensembles all obtained better results. CBE obtained a TPR of 84.27% an increase of 20.81% compared to VetaDetect. CBE with sliding window and HBE both showed an 100% detection rate, with a 36.54% increase compared to VetaDetect.

The results for the second comparison, on the Secondary Dataset II, between the proposed ensembles and the random forest based approach proposed by Roman et al. [24], are

summarized in Table 10 and the predictor configurations are summarized in Table 11.

In the tamper-free scenario, the proposed ensembles, namely, CBE, CBE with sliding window and HBE all yielded 0% FPR even with anonymized data. The random forest approach obtaining 18.5% FPR and 21.5% FPR on the anonymized version of the dataset. In comparison, this translating to a FPR decrease of 18.5% and 21.5%, respectively, obtained by the proposed ensembles.

Advancing to the first tampering-scenario, where for each predictor one signal was tampered. The best results were obtained by HBE and CBE with sliding window, with a TPR

TABLE 8. Performance comparison between VetaDetect [23] and the proposed detection ensembles.

Scenario	VetaDetect [23]		CBE		CBE Sliding Window		HBE	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Clean	-	0.0038	-	0	-	0	-	0
Emulator	1	-	0.8214	-	1	-	1	-
Intelligent ARX	0.9850	-	0.9952	-	1	-	1	-
Intelligent LSTM	0.6346	-	0.8427	-	1	-	1	-

TABLE 9. The complete list of selected inputs and outputs, for the LSTM predictive neural networks, on the datasets from [23], originating from a EURO VI D N2 class truck.

Predictor	Output	Input Variables
1	Intake NOx	Accelerator Pedal Position Engine Air Inlet Pressure Engine Fuel Rate Engine Torque Intake O ₂ Injected Urea
2	Outlet NOx	Intake NOx Part Filter Gas Temperature Engine Air Intake Pressure Part Filter Diff. Pressure Intake O ₂ Injected Urea
3	Outlet O ₂	Exhaust Gas Temperature Particle Filter Gas Temp. Engine Fuel Rate Engine Torque Intake NOx Intake O ₂ Injected Urea
4	Injected Urea	Accelerator Pedal Position Exhaust Gas Temperature Engine Fuel Rate Engine Throttle Position Exh. Part Filter Diff. Press. Intake NOx Intake O ₂

of 100%, while the point-by-point CBE obtained a 95.62% detection rate on the anonymized scenario and 95% on the non-anonymized scenario. The random forest approach obtained similar results, ranging from 76% when the Current of oxygen sensor was tampered all the way to 100% TPR, in the cases where the Coolant Temperature, Engine Torque, Throttle and Valve Position were tampered.

Lastly, the authors in [24], considered an advanced tampering scenario, where the tamperer simulates four concurrent signals (Current of Oxygen Sensor, Coolant Temperature, Engine Torque, Throttle and Valve Position). In this final scenario, the random forest approach obtained similar results to CBE with sliding window and HBE, namely, 100% TPR, even on the anonymized version of the dataset, while the point-by-point CBE obtained a 98.83% TPR on the anonymized version and 98.75% on the non-anonymized version of the dataset. This signifies an average of 1.21%

decrease in the TPR, compared to the other proposed ensembles and the random forest approach.

D. RESOURCE CONSUMPTION

Neural networks in general require higher resources to run. The LSTM-NNs from the proposed ensembles were implemented in the reference testbed from (ii), to observe how resource consumption increases while running concurrent predictors. While some predictors might use fewer inputs and a simpler architecture, this experiment measures the worst-case scenarios, meaning, all the LSTM-NNs predictors were configured with the maximum number of inputs from the ones proposed in Subsection IV-C. The final configuration of the LSTM-NNs includes 1 sequential layer with 10 input neurons, 1 hidden LSTM-NN layer with 10 LSTM-NN cells, 1 output regression layer with 1 neuron. As for the dataset used, the WHWC driving cycle was chosen.

1) CPU USAGE

The CPU usage was measured while concurrently running from 1 to 5 predictors, with increments of 1. At the start of the measurements, an idle state was measured to show the resource consumption with inactive predictors, since other processes were still running on the Raspberry PI. Figure 5 illustrates the actual CPU usage while running the experiments with an increasing number of predictors. Here, the predictors were idle for the first 20 seconds, running for 60 seconds, and idle for another 20 seconds. While running only one predictor, the CPU usage increased and remained almost constant at 25% ($\pm 0.1\%$), consequently, while running 2 or more predictors, the CPU usage increased to a maximum of 34% with an average of 31% for all experiments.

2) MEMORY USAGE

In a similar manner, the memory usage measurements considered the idle state first. Afterwards, 1 to 5 predictors were started in parallel, with increments of 1. Figure 6 illustrates the RAM usage with the entire dataset loaded in memory. It can be observed that the idle RAM usage was 3.8% (292 MB) and increased to 5.2% (400 MB) with one predictor, 5.3% (408 MB) with two predictors and 5.4% (416 MB) with 3-5 predictors. It is to be noted that in a real-world scenario the RAM usage might be lower as it wouldn't be necessary to load and hold the entire dataset in memory.

TABLE 10. Performance comparison between the random forest based approach proposed by Roman et al. [24] and the proposed detection ensembles.

Scenario	Roman et al. [24]		CBE		CBE Sliding Window		HBE	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Clean	-	0.1850	-	0	-	0	-	0
Clean Anonymized	-	0.2150	-	0	-	0	-	0
One Signal Tampered	[0.7740, 1]	-	0.9500	-	1	-	1	-
One Signal Tampered Anonymized	[0.7600, 1]	-	0.9562	-	1	-	1	-
Four Signals Tampered	1	-	0.9875	-	1	-	1	-
Four Signals Tampered Anonymized	1	-	0.9883	-	1	-	1	-

TABLE 11. The complete list of selected inputs and outputs, for the LSTM predictive neural networks, on the datasets from [24], originating from a 2015 EUR6 Skoda Rapid 1.2 L TSI passenger vehicle.

Predictor	Output	Input Variables
1	Coolant Temperature	Current of Oxygen Sensor Engine Speed Engine Torque Oxygen Jump Sensor Voltage Throttle Valve Position
2	Current of Oxygen Sensor	Coolant Temperature Engine Speed Engine Torque Oxygen Jump Sensor Voltage Throttle Valve Position
3	Engine Torque	Coolant Temperature Current of Oxygen Sensor Engine Speed Oxygen Jump Sensor Voltage Throttle Valve Position
4	Throttle Valve Position	Coolant Temperature Current of Oxygen Sensor Engine Speed Engine Torque Oxygen Jump Sensor Voltage

3) SCALABILITY

To get a sense of how scalable the proposed ensembles are, the experiments were extended to measuring the CPU and RAM usage while increasing the number of predictors from 1 to 20, on a single Raspberry PI device. The results, which include the minimum, maximum and average consumption are illustrated in Figure 7. In terms of CPU usage, the maximum value was 34%, while running 3 concurrent predictors and 33% with 20 concurrent predictors, while the average CPU usage was between 29.5% and 31.5% when running 2-20 concurrent predictors. The maximum RAM usage was 5.5% (424 MB), measured when running more than 17 predictors, consequently, the average measured RAM usage was between 5.3% (408 MB) and 5.4% (416 MB).

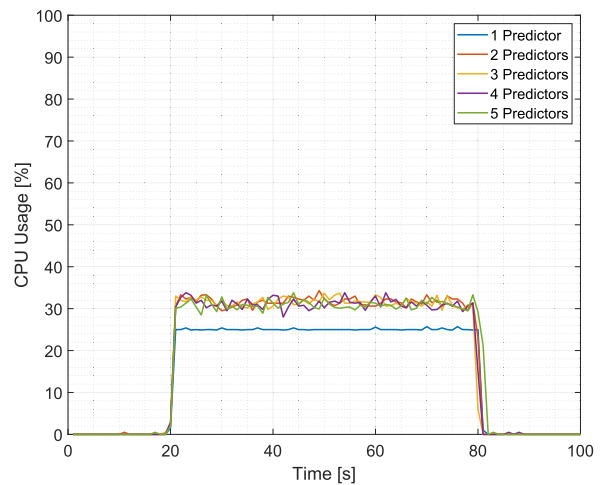


FIGURE 5. CPU Usage comparison when running 1-5 LSTM predictors on the Raspberry PI.

VI. DISCUSSIONS

The current paper presented results that point out the efficiency of the proposed tampering detection ensembles, together with resource measurements, that show the implementation possibility in real embedded systems. However, the following paragraphs will introduce the advantages and shortcomings of each ensemble, together with scenarios in which one can choose to implement an approach, or the other.

The first ensemble, using the Cumulative Sum (CUSUM) based detectors has several advantages. First, the CUSUM allows for the detection of small changes in the mean value and the standard deviation of the prediction error. If the CUSUM doesn't use a sliding window, then the detection starts instantly and has a higher frequency of detection (e.g., every data point). However, there will be a delay between the first anomalous data and the generation of an alert, as it was observed in Section V-B. This is owed to the fact that the CUSUM needs to accumulate enough information before surpassing the detection threshold. The threshold value can

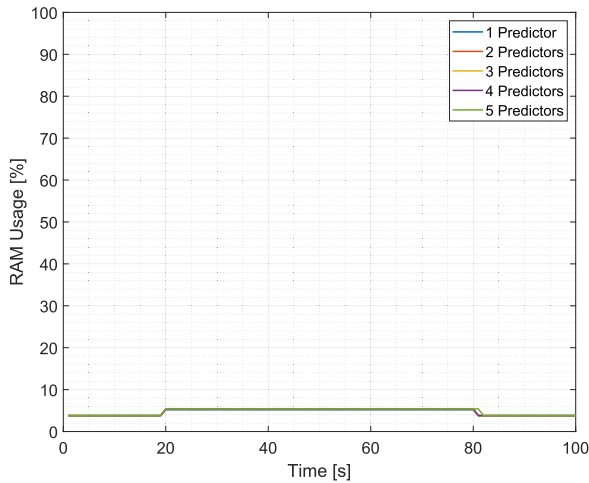


FIGURE 6. RAM Usage comparison when running 1-5 LSTM predictors on the Raspberry PI.

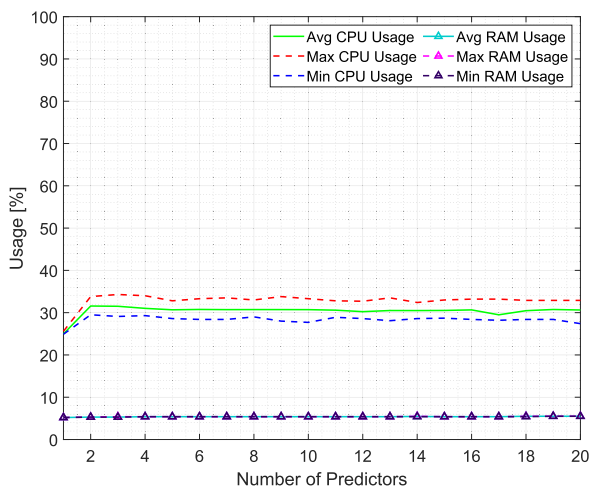


FIGURE 7. Average, Maximum and Minimum CPU and RAM usage when running 1-20 LSTM predictors on the Raspberry PI.

be adjusted, for a faster detection, this in term increasing the risk of false alerts. On the other hand, using a sliding window CUSUM approach, brings the advantage of resetting the cumulative sum to an initial zero value, after traversing the sliding window. The sliding window approach also offers the possibility of lowering the detection frequency by moving the sliding window with more than one point at a time, this is advantageous in the case of tampering, where the vehicle manipulations are persistent over extended periods of time.

The second ensemble, which uses the histogram distance based detectors, can monitor the changes in the distribution of the data. Even if the mean value or the standard deviation doesn't change, the distribution can still be affected by tampering. This method, compared to the point-by-point CUSUM approach, needs to accumulate sufficiently enough data points, which results in a delayed detection start. Nonetheless, as tampering does not cause obvious changes in

the distribution of the data, a larger number of data points is needed for detection. Furthermore, this method has a lower detection frequency, as one data point might not change the distribution of the data, thus the sliding window can be moved with a larger number of points. By having a lower frequency of detections, once every k points, also decreases the number of generated true/false alerts. This might be an advantage in environments with limited storage capabilities, or in environments where the alert is visible/audible, by not overwhelming the system and the user with alerts.

To get the best of two worlds, the detectors could be combined in an ensemble which uses both CUSUM based and histogram based detectors. This approach would have the advantage of being able to detect both the changes in the mean and variance but also the changes in the distribution of the prediction error. However, one would need to use two detectors for the same monitored signal (e.g., NO_x emissions, O₂, etc.), for a large number of monitored signals this, however, would increase the resource requirements. Moreover, the fusion system would have to be adjusted for the different frequencies at which the detectors work.

As documented in this paper, the design of the ensembles requires a data analysis procedure. Nonetheless, when implementing such a solution into a real environment, expert knowledge is needed to select the monitored signals and to finely tune the hyper-parameters. Moreover, the generated warnings and alerts should be analyzed. This can easily be done at periodical or at random road-side inspections. While the alerts would clearly indicate the presence of tampering in multiple components, the warning system could identify more subtle tampering, where the tamperer is able to alter only one component.

In a real-world scenario, the proposed detection methods are envisioned to function as ECU applications, with the possibility of distributing them among multiple ECUs, while the generated alerts would have to be stored in a secured environment, as not to be deleted or altered. Similarly to what the authors proposed in [23], the generated warnings and alerts could be aggregated, signed and securely stored using techniques such as secure logging, Trusted Platform Modules (TPMs) and Hardware Security Modules (HSMs).

VII. RELATED STUDIES

This section bisects in two distinct directions. Firstly, it analyses recent studies on tampering and anomaly detection in the automotive field, together with ensemble detection methods. Secondly, it presents related studies on the concept of TF for RNN to emphasize the approach of TF tackled in this paper.

A. AUTOMOTIVE TAMPERING DETECTION AND ENSEMBLE METHODS

In the automotive field, tampering detection is addressed by only a handful of papers. Nonetheless, in the scientific literature we find several other ensemble-based techniques addressing a large palette of issues, such as fault detection,

TABLE 12. Existing ensemble techniques applied in the automotive industry.

Author	Year	Ensemble Type	Base Models	Results Combination	Scope
Haller et al. [23]	2022	Independent	ARX	Dempster-Shafer	Tampering / Anomaly Detection
Wang et al. [54]	2020	Independent	Object Detection Models: Cascade-RCNN and RetinaNet	Soft-Weighted-Average	Autonomous Vehicle Object Detection
Theissler [55]	2017	Independent	One-class/Two-class classifiers	Majority Voting	Fault Detection
Sankavaram et al. [56]	2015	Independent	Support Vector Machine, k-Nearest Neighbor, Probabilistic Neural Network, Relevance Vector Machine	Weighted Voting	Fault Detection
Mozaffari and Azad [57]	2014	Independent	Extreme Learning Machines	Average of Outputs	Engine Coldstart Hydrocarbon Emission Identification

emission identification and object detection. The most relevant papers are summarized in Table 12.

One recent paper addressing tampering detection is the work of Haller et al. [23], where the authors proposed VetaDetect, an adaptive independent ensemble detection technique using Multiple-input Single-output (MISO) and Auto Regressive Moving Average (ARX) models. Their solution leverages the Granger causality for automated input/output model selection. Furthermore, the output from each individual model is fused using Dempster-Shafer (D-S) theory of evidence causality that offers a close-loop detection methodology, where the degree of belief for each detector is computed based on the reported belief on tampering. The authors provide an extended experimental assessment using two distinct ensemble configurations, namely, open-loop and closed-loop. In the open-loop scenario the confidence associated to each base detector is static, while in the closed-loop scenario it is automatically adjusted. Their dataset was created by the Vehicle Emissions Heavy Duty chassis laboratory at the Joint Research Centre of the EC, and it contains emission-related tamper-free data together with tampered data measured while using an AdBlue emulator mounted inside a real vehicle. A more detailed description of this dataset is presented in Section IV-A, as the study was used as a comparison reference in the experiments.

Locality Sensitive Hashing (LSH) was used as a methodology for detecting tampering in [58]. Here, the authors leverage the properties of LSH, namely, the high probability that two points close to each other in Euclidean space hash to similar values, to detect tampering in high dimensional data. The authors proved that even small deviations in the tampered data, produces high deviations in the LSH method, thus the probability of collision of the tampered data-points decreased significantly.

Roman et al. [24] proposed a novel privacy preserving tampering detection technique for automotive systems, well capable of detecting tampering even in anonymized data. They employed a Fast Fourier Transform (FFT) distortion method capable of preserving privacy in vehicle sensor data. While for tampering detection they used a Random Forest regressor together a custom sliding window CUSUM based approach. While the authors are not mentioning that their solution is an ensemble-based detection technique, Random Forest is in itself an ensemble method comprising multiple decision trees. Moreover, the tampering scenarios proposed in their paper includes a high number of tampering methods which could very well be carried off in future by malicious actors (e.g., tamperers). Section IV-A further describes this dataset, since similarly to [23], this work is a point of comparison.

Moving towards fault detection in the automotive field, we find the work of Theissler [55]. The paper describes an offline fault detection technique using an ensemble of two-class and one-class classifiers, trained using both clean and anomalous data. The classifier decisions are combined using a Weighted Voting methodology. To test their solution, the authors created a custom dataset. This dataset consisting of clean and faulty OBD-II measurements that were extracted from a Renault Twingo I passenger vehicle. Here, the authors used a self-made device that allowed them to manipulate the engine bay and inject faults ranging from ignition, engine temperature all the way to engine temperature faults.

Mozaffari and Azad [57] proposed an independent ensemble method using Extreme Learning Machines (ELM) for engine coldstart hydrocarbon emission identification. That is, modeling the behavior of a spark-ignited engines during coldstart operations. Their ensemble consists of several ELMs with different types of regularization techniques

(e.g., LASSO, Tikhonov, Elastic net, Cascade of LASSO and Tikhonov).

Another notable work is that of Taylor et al. [1]. Here, the authors designed a frequency-based anomaly detection technique for the Controller Area Network (CAN). Their solution aims to compute network traffic statistics (e.g., Hamming distance between packets and inter-packet timings), and compare them with historical values to produce an anomaly signal. Their study showcases experiments with one-class support vector machines for anomaly detection. To prove the efficiency of their method, the authors created a custom dataset simulating attack traffic using real CAN data collected from a 2011 Ford Explorer vehicle.

Longari et al. [59] designed an Intrusion Detection System (IDS) based on LSTM-NN auto-encoders to identify anomalies in CAN data. Their solution is automatically trained on CAN streams to create models of the legitimate data sequences. The reconstructed errors are later used for anomaly detection, by comparing them to the reconstructed errors computed during several simulated attacks. Similarly to [1], this solution is addressing anomaly detection at frame level, without decoding the actual signals contained in the payload of the data frames.

Other papers focused on clustering approaches to detect anomalies. Guerreiro et al. [60] offers a case study on the detection of pricing anomalies of different automotive parts that have similar physical characteristics. Detecting pricing anomalies would aid in optimizing the production costs of similar parts produced by different manufacturers. The case study applies and analyses multiple clustering algorithms (e.g., K-Means, Hierarchical clustering, or Fuzzy C-Means). These algorithms were ranked using the Borda count method [61]. In the final evaluation the Hierarchical method obtained the best results, closely followed by K-Medoids and K-Means. It is worth mentioning that their proposed solution is not envisioned as an in-vehicle anomaly detection technique, but rather as a tool for automotive manufacturers to optimize the production process and reduce their production costs.

B. TEACHER FORCING

Teacher Forcing (TF) was first proposed in [21] to avoid using Back Propagation Through Time (BPTT) because of the growing memory requirements for long training sequences. The authors of the original paper describe a training method that involves feeding back the current ground truth values as input in the subsequent time-steps. This way, forcing the neural network to remain as close as possible to the ground truth sequences. In recent years, several variants of TF have been proposed [62], [63], [64], [65], [66], most of them offering variations of this method for training RNNs.

Taigman et al. [62] proposed a neural Text-To-Speech method, based on a fixed size memory buffer, for mimicking voices based on audio samples captured in the wild. The authors describe a modified version of teacher forcing, where, during the training stage, the average between the ground

truth and the previous network output plus a random noise vector is fed to the input instead of just the ground truth value of the previous time step.

In the field of language modeling, the work of Drossos et al. [63] stands out. The authors offer a RNN based method for learning language models for sound event detection. Their method utilizes the ground truth values during the initial training epochs, and based on a probability, these values are gradually replaced by the predicted values of the model. Their approach is based on scheduled sampling [64]. Scheduled sampling is a method where during training a random decision is made on whether to use the previous ground truth value or the previous model prediction as input at the current step.

Lamb et al. [67] presented a Generative Adversarial Networks (GAN) approach titled *Professor forcing*. Their technique leverages an adversarial training approach that involves training an additional discriminator to differentiate between free-running and teacher-forced hidden states. This way they encourage the dynamics of the recurrent network to remain identical when using both previous ground truth and freely sampled values, during training.

In the field of anomaly detection, Loganathan et al. [65] present an anomaly detection technique for Transmission Control Protocol (TCP) requests. Their solution employs a multi-attribute prediction model for network packet sequences, using a Seq2seq encoder-decoder model to reduce the error propagation in testing. Their models are trained using the original version of teacher forcing. In a different direction, Massaoudi et al. [66] proposed a Photovoltaic Power Forecasting hybrid model, consisting of a combination between NARXNN and LSTM-NN. Their solution uses a NARXNN model to acquire data and generate a residual error vector which is fed as additional input to the LSTM-NN, which produces point-by-point and sequence forecasts.

The above studies leverage TF and use the trained models in a closed loop forecasting mode (e.g., the network predicts future time-steps by also using the previous predictions as input). In this case, the model does not require the ground truth values to make a prediction. In comparison, the predictors proposed in this paper are used in an open loop forecasting mode, where they predict future time-steps by using the input data together with the ground truth response value from the previous time-step. The detectors monitor certain signals by constantly analyzing the deviations (e.g., errors), between the ground truth value and the predicted value, using two distinct methodologies (e.g., CUSUM and Histogram analysis). To the best of my knowledge, this is the first proposed LSTM-NN based tampering (or anomaly) detection solution which utilizes TF, with the ground truth value, from the previous time-step, being fed back as input, during both training and detection (inference).

VIII. CONCLUSION

This paper addressed a new emerging threat in the automotive field, namely, tampering of the vehicles environmental protection systems. Tampering can have serious effects on both

human health and the environment, as tampered vehicles emit higher concentration of pollutants, such as nitrogen oxides and particulate matter. In response to this threat, this paper proposed two ensemble-based methodologies for tampering detection. The proposed solutions are concurrently using LSTM predictive neural networks together with CUSUM and histogram distance-based detectors. The CUSUM and histogram distance-based detectors receive as input the prediction error from the predictive models and output a binary decision using a threshold-based approach. Additionally, this paper introduced an adaptive majority weighted voting fusion methodology which considers the past decisions of the detectors in the weight adjustment procedures.

The proposed ensembles obtained notable results, over a large number of tampering methods and scenarios, originating from three different datasets, even in comparison with state-of-the-art tampering detection solutions. Furthermore, the possibility of integrating the proposed techniques in real embedded environments was also demonstrated. As future work, the ensembles will be further refined, while exploring the possibility of using different components (e.g., predictors, detection methods). Additionally, the proposed solutions will be tested on datasets from other domains, and on other types of attacks.

REFERENCES

- [1] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.
- [2] M. Weber, S. Klug, E. Sax, and B. Zimmer, "Embedded hybrid anomaly detection for automotive can communication," in *Proc. 9th Eur. Congr. Embedded Real Time Softw. Syst. (ERTS)*, 2018. [Online]. Available: https://hal.inria.fr/ERTS2018/search/index/?q=Embedded+hybrid+anomaly+detection+for+automotive+can+communication&submit=&sort=producedDate_tdate+desc
- [3] T. Lenard and R. Bolboacă, "A statefull firewall and intrusion detection system enforced with secure logging for controller area network," in *Proc. Eur. Interdiscipl. Cybersecurity Conf.*, Nov. 2021, pp. 39–45.
- [4] M. D. Pesé, K. Schmidt, and H. Zweck, "Hardware/software co-design of an automotive embedded firewall," SAE, Warrendale, PA, USA, Tech. Rep. 2017-01-1659, 2017.
- [5] Y. Laarouchi, M. Kaâniche, V. Nicomette, I. Studnia, and E. Alata, "A language-based intrusion detection approach for automotive embedded networks," *Int. J. Embedded Syst.*, vol. 10, no. 1, p. 1, 2018.
- [6] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.
- [7] T. Lenard, R. Bolboacă, B. Genge, and P. Haller, "MixCAN: Mixed and backward-compatible data authentication scheme for controller area networks," in *Proc. IFIP Netw. Conf. (Networking)*, 2020, pp. 395–403.
- [8] T. Lenard, R. Bolboacă, and B. Genge, "LOKI: A lightweight cryptographic key distribution protocol for controller area networks," in *Proc. IEEE 16th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2020, pp. 513–519.
- [9] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (CAVs)," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6240–6259, Jul. 2022.
- [10] A. Singh and M. Singh, "An empirical study on automotive cyber attacks," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 47–50.
- [11] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, Apr. 2019.
- [12] G. Baldini, R. Giuliani, and M. Gemo, "Mitigation of odometer fraud for in-vehicle security using the discrete Hartley transform," in *Proc. 11th IEEE Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UMCCON)*, Oct. 2020, pp. 0479–0485.
- [13] S. Terzi, C. Savvaids, K. Votis, D. Tzovaras, and I. Stamelos, "Securing emission data of smart vehicles with blockchain and self-sovereign identities," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Nov. 2020, pp. 462–469.
- [14] J. van den Meiracker and R. Vermeulen. (Dec. 2020). Status quo of critical tampering techniques and proposal of required new OBD monitoring functions. DIAS Project, Deliverable 3.2. Accessed: Sep. 13, 2022. [Online]. Available: https://dias-project.com/Deliverables/All_WPs
- [15] J. A. van den Meiracker and R. Vermeulen. (2020). Deliverable 3.1: The market of cheating devices and testing matrix with a prioritization for testing of vehicle tampering technique combinations. DIAS project deliverable: Smart Adaptive Remote Diagnostic Antitampering Systems. Accessed: Sep. 13, 2022. [Online]. Available: https://dias-project.com/Deliverables/All_WPs
- [16] DIAS. *Diagnostic Anti-Tampering Systems*. Accessed: Sep. 1, 2022. [Online]. Available: <https://Dias-Project.Com>
- [17] I. Ertug, "Motion for a European parliament solution with recommendations to the commission on odometer manipulation in motor vehicles: Revision of the EU legal framework," Report of the European Parliament, Committee on Transport and Tourism, Brussels, Belgium, Tech. Rep. 2.5.2018, 2018. [Online]. Available: https://www.europarl.europa.eu/doceo/document/A-8-2018-0155_EN.html
- [18] M. O. Environment and F. O. Denmark, "Measurements of cheating with SCR catalysts on heavy duty vehicles," Environmental Protection Agency, Washington, DC, USA, Environ. Project no. 2021, 2018. [Online]. Available: <https://www2.mst.dk/Udgiv/publications/2018/06/978-87-93710-42-9.pdf>
- [19] B. Giechaskiel, F. Forloni, M. Carriero, G. Baldini, P. Castellano, R. Vermeulen, D. Kotses, P. Fragkiadoulakis, Z. Samaras, and G. Fontaras, "Effect of tampering on on-road and off-road diesel vehicle emissions," *Sustainability*, vol. 14, no. 10, p. 6065, May 2022. [Online]. Available: <https://www.mdpi.com/2071-1050/14/10/6065>
- [20] C. Braun, H. Badshah, V. Hosseini, L. Jin, J. Miller, and F. Rodríguez, "Heavy-duty emissions control tampering in Canada," in *Proc. Int. Council Clean Transp. (ICCT) Rep.*, 2022, pp. 1–65.
- [21] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, Jun. 1989.
- [22] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1191–1214, 1990.
- [23] P. Haller, B. Genge, F. Forloni, G. Baldini, M. Carriero, and G. Fontaras, "VetaDetect: Vehicle tampering detection with closed-loop model ensemble," *Int. J. Crit. Infrastructure Protection*, vol. 37, Jul. 2022, Art. no. 100525. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1874548222000154>
- [24] A.-S. Roman, B. Genge, A.-V. Duka, and P. Haller, "Privacy-preserving tampering detection in automotive systems," *Electronics*, vol. 10, no. 24, p. 3161, Dec. 2021.
- [25] J. K. Andrea Strzelec, *Automotive Emissions Regulations and Exhaust Aftertreatment Systems*. Warrendale, PA, USA: SAE International, 2020.
- [26] M. Chen and A. A. Ghorbani, "A survey on user profiling model for anomaly detection in cyberspace," *J. Cyber Secur. Mobility*, vol. 8, no. 1, pp. 75–112, 2019.
- [27] H. Izakian and W. Pedrycz, "Anomaly detection in time series data using a fuzzy C-means clustering," in *Proc. Joint IFSA World Congr. NAFIPS Annu. Meeting (IFSA/NAFIPS)*, Jun. 2013, pp. 1513–1518.
- [28] C. D. Truong and D. T. Anh, "An efficient method for motif and anomaly detection in time series based on clustering," *Int. J. Bus. Intell. Data Mining*, vol. 10, no. 4, pp. 356–377, 2015.
- [29] M. Bulygin and D. Namiot, "Anomaly detection method for aggregated cellular operator data," in *Proc. 28th Conf. Open Innov. Assoc. (FRUCT)*, Jan. 2021, pp. 42–48.
- [30] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*. Boca Raton, FL, USA: CRC Press, 2014.
- [31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [32] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*. New York, NY, USA: Springer, 1998, pp. 199–213.

- [33] R. C. Staudemeyer and E. Rothstein Morris, "Understanding LSTM—A tutorial into long short-term memory recurrent neural networks," 2019, *arXiv:1909.09586*.
- [34] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D, Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.
- [35] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
- [36] Z. Wu and Q. Wang, "A single CUSUM chart using a single observation to monitor a variable," *Int. J. Prod. Res.*, vol. 45, no. 3, pp. 719–741, Feb. 2007, doi: [10.1080/00207540600792267](https://doi.org/10.1080/00207540600792267).
- [37] E. GmbH. *Exothermia Suite*. Accessed: Sep. 5, 2022. [Online]. Available: <https://www.exothermia.com/exhaust-overview>
- [38] R. Bolboacă, P. Haller, D. Kontses, A. Papageorgiou-Koutoulas, S. Doulgeris, N. Zingopis, and Z. Samaras, "Tampering detection for automotive exhaust aftertreatment systems using long short-term memory predictive networks," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroSPW)*, Jun. 2022, pp. 358–367.
- [39] *World Harmonized Vehicle Cycle (WHVC)*. Accessed: Sep. 5, 2022. [Online]. Available: <https://dieselnet.com/standards/cycles/whvc.php>
- [40] *European Transient Cycle (ETC)*. Accessed: Sep. 5, 2022. [Online]. Available: <https://dieselnet.com/standards/cycles/etc.php>
- [41] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981.
- [42] G. R. Terrell and D. W. Scott, "Oversmoothed nonparametric density estimates," *J. Amer. Stat. Assoc.*, vol. 80, no. 389, pp. 209–214, Mar. 1985.
- [43] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.
- [44] H. A. Sturges, "The choice of a class interval," *J. Amer. Stat. Assoc.*, vol. 21, pp. 65–66, Mar. 1926.
- [45] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *Proc. 35th German Conf. Artif. Intell.*, vol. 9, 2012, pp. 1–5.
- [46] E.-S. Hamza, A. Abou El Kalam, A. Outchakoucht, and S. Benhadou, "Machine learning enhanced access control for big data," *Int. J. Comput. Sci. New. Secur.*, vol. 20, no. 3, p. 83, 2020.
- [47] H. Es-Samaali, A. Outchakoucht, S. Benhadou, O. Mounnan, and A. Abou El Kalam, "Anomaly detection for big data security: A benchmark," in *Proc. 3rd Int. Conf. Big Data Eng. Technol. (BDET)*, Jan. 2021, pp. 35–39.
- [48] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann, "Empirical evaluation of dissimilarity measures for color and texture," *Comput. Vis. Image Understand.*, vol. 84, no. 1, pp. 25–43, Oct. 2001.
- [49] P. A. Marín-Reyes, J. Lorenzo-Navarro, and M. Castrillón-Santana, "Comparative study of histogram distance measures for re-identification," 2016, *arXiv:1611.08134*.
- [50] D. Lane, D. Scott, M. Hebl, R. Guerra, D. Osherson, and H. Zimmer, *Introduction to Statistics*. Princeton, NJ, USA: Citeseer, 2003.
- [51] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
- [52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [53] R. Giampaolo. (2020). *Psutil: Cross-Platform Lib for Process and System Monitoring in Python*. Accessed: Aug. 17, 2022. [Online]. Available: <https://github.com/giampaolo/psutil>
- [54] H. Wang, Y. Yu, Y. Cai, X. Chen, L. Chen, and Y. Li, "Soft-weighted-average ensemble vehicle detection method based on single-stage and two-stage deep learning models," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 1, pp. 100–109, Mar. 2021.
- [55] A. Theissler, "Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection," *Knowl.-Based Syst.*, vol. 123, pp. 163–173, May 2017.
- [56] C. Sankavaram, A. Kodali, K. R. Pattipati, and S. Singh, "Incremental classifiers for data-driven fault diagnosis applied to automotive systems," *IEEE Access*, vol. 3, pp. 407–419, 2015.
- [57] A. Mozaffari and N. L. Azad, "Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine coldstart hydrocarbon emission identification," *Neurocomputing*, vol. 131, pp. 143–156, May 2014.
- [58] R. Bolboacă, T. Lenard, B. Genge, and P. Haller, "Locality sensitive hashing for tampering detection in automotive systems," in *Proc. 15th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1–7.
- [59] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero, "CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2021.
- [60] M. T. Guerreiro, E. M. A. Guerreiro, T. M. Barchi, J. Biluca, T. A. Alves, Y. de Souza Tadano, F. Trojan, and H. V. Siqueira, "Anomaly detection in automotive industry using clustering methods—A case study," *Appl. Sci.*, vol. 11, no. 21, p. 9868, Oct. 2021.
- [61] P. Emerson, "The original Borda count and partial voting," *Social Choice Welfare*, vol. 40, no. 2, pp. 353–358, 2013.
- [62] Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani, "VoiceLoop: Voice fitting and synthesis via a phonological loop," 2017, *arXiv:1707.06588*.
- [63] K. Drossos, S. Gharib, P. Magron, and T. Virtanen, "Language modelling for sound event detection with teacher forcing and scheduled sampling," 2019, *arXiv:1907.08506*.
- [64] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.
- [65] G. Loganathan, J. Samarabandu, and X. Wang, "Sequence to sequence pattern learning algorithm for real-time anomaly detection in network traffic," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2018, pp. 1–4.
- [66] M. Massaoudi, I. Chihli, L. Sidhom, M. Trabelsi, S. S. Refaat, H. Abu-Rub, and F. S. Oueslati, "An effective hybrid NARX-LSTM model for point and interval PV power forecasting," *IEEE Access*, vol. 9, pp. 36571–36588, 2021.
- [67] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.



ROLAND BOLBOACĂ received the bachelor's degree in informatics and the M.Sc. degree in information technology from the George Emil Palade University of Medicine, Pharmacy, Science and Technology of Târgu Mureș, Romania, where he is currently pursuing the Ph.D. degree in the field of computer science. He is also an Assistant Researcher at the George Emil Palade University of Medicine, Pharmacy, Science and Technology of Targu Mures. His research interests include anomaly detection, data analysis, machine learning, and security systems.