Received 19 April 2023; revised 9 June 2023; accepted 11 December 2023. Date of publication 18 January 2024; date of current version 26 January 2024.

*Digital Object Identifier 10.1109/OJITS.2023.3347484*

# Fault Prediction and Recovery Using Machine Learning Techniques and the HTM Algorithm in Vehicular Network Environment

**SALAH ZIDI[1,2,3], BECHIR ALAYA[1,2,3,4], TAREK MOULAHI[5,6], AMAL AL-SHARGABI[5], AND SALIM EL KHEDIRI[5,7]**

[1]Electrical Department, Gabes University, Gabes 6029, Tunisia

[2]Higher Institute of Industrial Systems, Gabes University, Gabes 6029, Tunisia

[3]Hatem Bettaher Laboratory, Gabes University, Gabes 6029, Tunisia

[4]Department of Management Information Systems and Production Management, College of Business and Economics, Qassim University, Buraidah 52571, Saudi Arabia

[5]Department of Information Technology, College of Computer, Qassim University, Buraydah 52571, Saudi Arabia

[6]FSTSBZ, Kairouan University, Kairouan 3100, Tunisia

[7]Department of Computer Science Faculty of Sciences of Gafsa, University of Gafsa, Gafsa 2112, Tunisia

CORRESPONDING AUTHOR: S. EL KHEDIRI (e-mail: salim.el.khediri@gmail.com)

**ABSTRACT** The amount of data available to vehicles has become very large in the vehicular networks' environment. Failures that mislead real-time data from vehicle sensors and other devices have become massive, and the need for automated techniques that can analyze data to detect malicious sources has become paramount. The application of machine learning techniques in the environment of vehicular ad hoc networks (VANET) is very promising and is beginning to show results in terms of applications designed and articles published. These techniques are increasingly accessible and used intensively, as many researchers are working to detect anomalous data. However, there is no universal, effective technique so far that can detect all abnormal data and then recover it. This work is an effort in that direction. We propose a smart model that uses multiple machine-learning classification methods. Our contribution also relates to a study of the attributes of interest for the algorithm used during the detection phase, namely the hierarchical temporal memory algorithm (HTM). The packets exchanged by the vehicle are grouped in instant description windows. These windows are then analyzed to extract a set of attributes. These are linked to the properties of network traffic such as flow or latency. They are subject to the process of detecting anomalies and intrusions carried out thanks to the algorithm with HTM. We propose the performance of fault detection and recovery at the level of the fog layer. The obtained simulation results demonstrate the efficiency of the learning methods and HTM for the detection of defects and errors in the IoV.

**INDEX TERMS** Vehicular network, fault prediction, fault recovery, Internet of Vehicles (IoV), machine learning, hierarchical temporal memory (HTM), classification.

## I. INTRODUCTION

LET'S assume a scenario where, you are sending information through the vehicular ad hoc networks (VANET), but you are not getting a satisfactory response even within the best results. These are called data faults or communication failures.

VANETs are becoming an important source of preventing vehicle accidents by improving road safety, traffic control, and passenger comfort [2], [3]. The reason is that people are using smart vehicles more frequently than before.

The review of this article was arranged by Associate Editor Anand Paul.

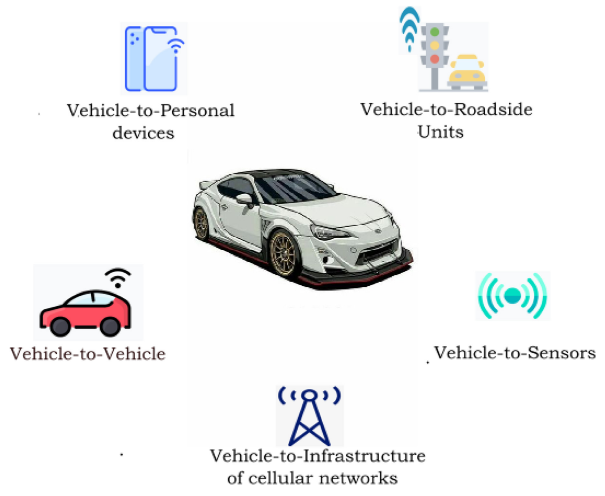
Vehicle-to-Personal devices
Vehicle-to-Roadside Units
Vehicle-to-Vehicle
Vehicle-to-Sensors
Vehicle-to-Infrastructure of cellular networks

FIGURE 1. IoV involve five types of communications, Vehicle to Vehicle, Vehicle to Sensors, Vehicle-to-Roadside, Vehicle to Personal devices, and Vehicle to cellular networks infrastructure.

Nevertheless, there are still several challenges in VANETs such as unreliable Internet service, limited processing capacity, incompatibility with personal devices, lack of cloud computing services, etc.

Internet of Things (IoT) is a technology of connects the physical and digital worlds. IoT has enabled communications to be more informative, the processing to be faster, and devices smarter [10]. This emerging technology has led to anytime, anywhere, anyway, anything communication. In intelligent transport systems (ITS), one of the most active research fields is the Internet of Vehicles (IoV) which apply IoT in VANETs [11]. IoV involve five types of communications (see Fig. 6) [11]: (1) Vehicle to Vehicle, (2) Vehicle to Sensors, (3) Vehicle to Roadside Units, (4) Vehicle to Personal devices, and (5) Vehicle to cellular networks infrastructure.

Failures may occur at any type of communication mentioned above. The failures that may threaten the IoV can be arranged into two classes [5]: vehicle-based failures and infrastructure-based failures. The vehicle-based failures can be hardware, software, mechanical, communication system, or failure related to interaction platform failure, e.g., the system is not able to detect command. Infrastructure-based failures involve failures related to other road users, failure due to weather impact, failures due to construction zones, failures due road conditions like failures related to traffic signals and signs.

As the Internet of Vehicles (IoV) is mainly based on Wireless Sensor Networks (WSNs), sensors are usually installed in an unattended and unfriendly environment, which in turn may lead to severe faults. To avoid such risky situations, this study proposes a joint method for fault detection and recovery with the help of machine learning approach. In particular, we propose four classification methods for detection namely, and a recovery method based on an aggregation approach executed at fog level. To the best of our knowledge, in our work this is the first study that combines both detection and recovery in a single study in the filed of IoV.

Leaving from the hypothesis that modern vehicles are equipped with a set of sensors and actuators connected by an internal network. The information that this equipment generates is processed by electronic control units (ECU) in order to allow vehicles to make complex decisions such as detecting and avoiding obstacles. Consequently, the information is collected mainly by the vehicles which regularly share the interpretation of its environment as well as its state with a distant interlocutor.

Data verification is one of the essential aspects of transportation systems in general and IoV in specific, which should be primarily focused on. As IoV relies on real-time information, data needs to be verified and error-free. Faults and errors in data can cause disastrous scenarios. In IoV networks, it is very possible that malicious nodes can exist and change the information sent by the sender. Thus, frequent data verification is highly required to prevent data alteration or false information [11]. This research focuses on data integrity/data trust security requirement by improving the verification of data generated and transmitted in IoV. The data verification is accomplished detecting several types of faults and in early stages. There is a need to propose an alternative approach to create a more reliable IoV environment.

The use of machine learning methods tends to be an attractive approach to detect errors in data generated in IoV. These learning methods enables automated learning, recognition of complex models, and intelligent decision making. There various types of machine learning methods, however, classification looks the most appropriate and efficient method for fault and errors detection [12]. It is used for decision-making assistance and data categorization. Moreover, this study is motivated by the lack of studies that combine both fault detection and recovery in a single study. Besides, the existing studies do not consider the IoV computation layers, i.e., edge, fog, and cloud and thus we propose that the performance of detection and recovery of faults at the fog layer.

In this paper, the main contributions are (i) proposal of a smart model that uses four classification methods to detect faults in IoV, (ii) the suggestion of an aggregation-based method for correcting the IoV fault. and finally (iii) performing fault detection and recovery at the fog computing layer which enables early fault discovery and correction.

The rest of the paper is organized as follow. Section II outlines a literature review. In Section III, the theoretical base of the model is given. The proposed model of fault detection and recovery in IoV is given in Section IV. Section V presents and discusses the experimental study. Finally, the conclusion is given in Section VI.

## II. RELATED WORK

To fight against DOS attacks in a vehicular network, several solutions and algorithms have been proposed and studied in the literature. These solutions present ideas for detecting and stopping these attacks.

In [40], the authors proposed the RRDA algorithm (Request Response Detection Algorithm) to detect the DOS attack (Denial of Service) in the VANET network. The authors assume that all nodes are equipped with an ORT (Onboard Radio Transponder) which decides and defines the vehicles which can form a network according to their transmission range. This is considered a threshold. Vehicles can make a request to the RSRT (Road Side Radio Transducer) to join the network created using the APDA (Attack Packet Detection Algorithm) mechanism. The RSU uses its own database of requests and responses and provides services only to those ORTs that are already verified, reducing the DOS attack. In the study [41], the authors proposed the APDA algorithm (Attacked Packet Detection Algorithm) to detect the DOS attack. This algorithm minimizes processing delay and improves security in the VANET network. Each vehicle has an OBU and a Tamper Proof Device. These devices store detailed vehicle information (speed, position, etc.).

Herrera et al. [42] proposed the RBS protocol model (Reference Broadcast Synchronization) for the prevention of DOS attacks in the VANET network. This model is based on the "Master Chock Filter" concept for filtering packets while traffic is heavy or during other attacks. RBS protocol has shown that packet delivery rate, throughput, timing are improved compared to IP-trackback protocol.

The authors of [43] discuss several cryptographic solutions for several possible attacks on the VANET network, including two solutions for "Jamming" and DOS attacks on network availability. For the first solution studied in [44], the authors propose to change the transmission channel and to use the FHSS technique (Frequency Hopping Spread Spectrum) which involves cryptographic algorithms to generate pseudo-random numbers for the hopping algorithm (FHSS). The second solution is the same solution proposed in the study of [45]. The proposal is to use "Signature based authentication mechanisms" a mechanism to reduce the effect of the DOS attack.

In the study [46], the authors propose an innovative probabilistic model based on logistic regression. This method makes it possible to estimate the appearance of an attack. The method is based on a database that estimates attack occurrences. When the regression model is validated, it is used to estimate the probability of an attack. If this probability exceeds the threshold defined in advance, the attack is therefore confirmed.

Muhammed and Shaikh [9] proposed a classification taxonomy of fault detection techniques in WSNs, which contains three main categories: centralized, distributed, or hybrid. In the centralized approaches, there is a single primary node that analyzes the data coming from the remaining nodes. The distributed approaches rely on local analysis in which the analysis is dispersed on all the nodes in the network. The hybrid approaches combine the aspects of centralized and distributed techniques. In the context of this research, the algorithms used in centralized, distributed, and hybrid approaches techniques can be further identified to be either statistical-based or machine learning-based.

Traditionally, statistical methods such as Probabilistic-based [14] and Threshold-based and account-based [15] are used to detect faults types in WSNs. However, such methods have some disadvantages, such as the uncertainty in the selection of the threshold values and how often the detection procedure should be repeated [15]. Thus, machine learning methods are promising in detecting the faults that may exist in WSNs. An overview of the studies performed for diagnosing faults detection based on the statistical and machine learning methods are presented with an emphasis on the latter.

### A. STATISTICAL BASED FAULT DETECTION TECHNIQUES

Panda et al. [16] proposed a wireless sensor network algorithm for detecting faults based on the z-value method. In this study, the researchers considered offset and only gain faults. Using false alarm rates and detection accuracy performance measures, their simulation results outperformed the conventional statistical algorithms.

Panda et al. [17] suggested an autoregressive model to diagnose faults. To validate the proposed model, an experiment was conducted on various sensor data that were generated under different conditions. The experimental results showed that the autoregressive model could successfully distinguish between the WSNs normal data and faulty data.

Hornik et al. [18] proposed a statistical algorithm for the detection of soft as well as hard faults. This algorithm is called the modified three-sigma edit test. The experimental results of this study showed the proposed algorithm outperformed the traditional methods in detection accuracy, as well as for false alarm rate and false-positive rate.

Panda and Khilar [19] suggested a new distributed detection scheme for sensor networks. The proposed solution was based on a method known as the error function. The proposed approach used majority voting, in which the sensor node takes a decision based on a comparison between its own sensing data from neighbors' data. The use of error function helped to increase the detection accuracy of faults.

Yuan et al. [21] proposed a novel algorithm called Distributed Bayesian Algorithm (DBA) for faulty nodes detection. The algorithm consists of 3 steps. Step one is to calculate the probability of the faults by exchanging readings between the sensor and neighbor nodes. In the second step, the probability calculated in the first step is validated and modified if it was not correct. In the third step, a warning message is sent to the base station if the sensor node's fault probability exceeds the threshold's probability.

The proposed algorithm was compared with the existing approach using an experiment, and the results showed that the algorithm outperformed them, especially with a big number of neighbors, and with a high rate of faults.

### B. MACHINE LEARNING-BASED FAULT DETECTION TECHNIQUES

The use of ML is an attractive approach for fault detection in WSNs. Many researchers have proposed different machine learning methods for fault detection in WSNs. Among the common machine learning methods used are Hidden Markov Model (HMM) [22], Support Vector Machines [23], [29], Neural Networks and Deep Learning [16], [24], Naïve Bayes [25], [26], K-Nearest Neighbors [27].

Warriach and Tei [22] applied the HMM, which is a supervised learning algorithm for fault diagnosis in WSN. The considered offset, gain, and stuck-at faults, and they compared HMMs of faulty and non-faulty environments. The model was evaluated using real-world data to was able to detect offset, gain, stuck-at faults with accuracies of 93.47, 94.02, 94.03, respectively.

Karmarkar et al. [23] proposed a support vector machine model to detect faults The model was tested to be able to effectively increase positive fault rate, fault alarm rate, and fault detection accuracy. The SVM features were optimized by Grey Wolf optimizer, and the model was compared with existing approaches of fault detection. Also, in a previous work of ours [29], we proposed an SVM-based model to detect faults in a dataset that was prepared by us to contain several types of faults. Our experiment results showed that the model is useful in predicting faults in the prepared dataset.

Swain and Khilar [24] have proposed a fault detection algorithm to distinguish various kinds of faults. Examples include hard and soft permanent, intermittent, and transient. More recently, based on deep learning methods, Panda et al. [16] proposed a model to diagnose the intermittent faults in wireless sensor network. The performance of the diagnosis method was measured by false-positive rate, detection accuracy, and false alarm rate. The proposed model of their study achieved 100% detection accuracy when 30 numbers of data from the sensor used and the intermittent fault probability are more than 0.25.

Lau et al. [25] proposed a new algorithm, namely, Centralized Naıve Bayes Detector (CNBD), the is based on the Naıve Bayes method to detect hardware faults. In the proposed algorithm, a new attribute, which is the transmission time of the arrived packet at the sink is analyzed using the Naıve Bayes method. Also, Warriach and Tei [22], in their study, classified the sensed data into normal or outlier. The classification is based on the thermos of Bayes [28]. The proposed method operates in two levels: sensor node and cluster-head levels and the Naıve Bayes classifier is used at the first level. The model performance was tested on a real and a synthetic dataset and achieve an accuracy performance of 0.88.

Yadav and Ahamad [27] proposed an approach for outlier detection using the KNN prediction model. The algorithm was tested on a real-time dataset generated by volunteers from the Intel Berkeley lab. The model achieved a detection accuracy f 0:86. Talking a closer look at the studies mentioned above, it can be observed that most of the works have applied a limited number of machine learning algorithms, mostly one or two, to detect faults. In contrast, this study proposes the use of multiple machine learning methods, i.e., five methods, to detect different types of faults and also recover data in the WSNs.

## III. PROPOSED FAULT DETECTION AND RECOVERY MODEL

The main idea of the proposed model is to apply machine learning classification methods and the hierarchical temporal memory (HTM) algorithm initially presented in "On Intelligence" by Hawkins and Blakeslee [47], to detect several types of defaults. HTM has already demonstrated good capacities for learning sequences [48], but also for detecting faults in temporal or sequential data [49], [50], [51]. Classification should be implemented at the fog layer of IoV. Therefore, in the following subsections, we discuss the different types of errors detected, the classification methods used, the HTM algorithm and those many properties that make it suitable for its use in our context of fault detection in vehicular networks, and finally a brief description of the IoV layers with emphasis on the fog layer.

### A. TYPES OF FAULTS DETECTED IN THE PROPOSED MODEL

As mentioned earlier in the introduction section, failures that may threaten the IoV can be categorized into two groups: failures related to vehicles and failures related to infrastructure [5]. These two groups of failures can appear due to the collected data faults or due to the hardware faults in the IoV network. In this study, we focus on data-related faults. In general, collected data from IoV network can be represented as a triplet $d(n, t, f(t))$; Where $f(t)$ is the collected data by the node n throughout $t$. $f(t)$ can be represented as the following equation:

$$\alpha + \beta x + \eta \tag{1}$$

where $\alpha$ represents the offset, $\beta$ is the gain, $x$ is the gathered data by the node $n$ throughout $t$, and $\eta$ is the noise that exists in the collected data [5]. In this study, we consider five types of data-related faults. These types are explained below.

#### 1) DATA LOSS FAULT

This type of fault refers to the missing values of gathered data by a specific node during certain periods or series. That is, collected data hold null values. Data loss fault is represented by the following equation:

$$f(t) = \phi, t > \tau \tag{2}$$

$\tau$ is representing the most required extreme time to sense information, and $\phi$ is representing a null set.

### 2) STUCK-AT FAULT

In this type, the variation of the collected data within a period of time is equal to zero. Stuck-at fault can be represented as follows:

$$x' = \alpha \tag{3}$$

$\alpha$ is representing the data sensed by the node $n$ at the time $t$.

### 3) SPIKE FAULT

In this type, the change rate of actual time series and anticipated time arrangement exceeding the normal changing pattern. Spike fault can be represented as shown below:

$$\frac{|f(t) - f^p(t)|}{t} > \lambda \tag{4}$$

$f(t)$ is representing the actual data, $f^p(t)$ is the anticipated time arrangement at a time $t$, while $\lambda$ indicates the actual changing trend in the gathered information.

### 4) OUT OF BOUNDS

In this type, the gathered data are beyond the expected data range. Let's assume that the expected data range is the interval $[\theta_1, \theta_2]$, a fault is out of bounds when the collected data $x' \in f(t)$ such that:

$$x' < \theta_1 \ or \ x' > \theta_2 \tag{5}$$

### 5) GAIN FAULT

In this type, the actual reading of the data may be increased by a particular value. This may happen when rate of data change is larger than the familiar reading. Gain fault can be defined by:

$$x' = \beta x + \eta \tag{6}$$

where $\beta$ represents the gain value to be multiplied with the actual reading $x$.

### 6) OFFSET FAULT

In this type, the actual reading of the data may be displaced by a certain value, i.e., offset. This may happen due to a wrong calibration in the sensor device. This type of fault can be represented by the following equation:

$$x' = \alpha + x + \eta \tag{7}$$

$\alpha$ indicates the offset rate added to the actual measurement $x$.

## *B. MACHINE LEARNING CLASSIFICATION METHODS AND HIERARCHICAL TEMPORAL MEMORY ALGORITHM USED IN THE PROPOSED MODEL*

The model suggests the use of machine learning classification methods for detecting faults. In the context of this research, the classification methods are used to predict whether received data from the IoV sensors are normal or faulty. Four classifiers are applied and compared in terms of performance: multilayer perceptron, support vector machines, decision trees, and random forest. The fault detection process that we have designed is based on the use of an unsupervised detection algorithm. Therefore, the choice of the algorithm was motivated by the following aspects: the impact on the performance, the load and the nature of the traffic in real-time, and the diversity in the temporal properties of the faults that we wish to detect, the autonomy in learning the algorithm. Various methods are commonly used for autonomous fault detection in the VANET environment [52], [53], [54]. However, these methods all have one thing in common: when the unsupervised learning phase of the algorithm ends, it is no longer able to learn new things. Therefore, if the traffic were to change, the detection would be less and less reliable and the algorithm would have to be re-initialized in order to adapt to the new traffic. To our knowledge, the only unsupervised algorithm capable of continuously learning and adapting to the evolution of traffic is the hierarchical temporal memory (HTM) algorithm.

### 1) PROPERTIES OF HIERARCHICAL TEMPORAL MEMORY (HTM) ALGORITHM

The unique context of vehicular networks leads us to consider the use of a detection method by the nature of the traffic and the entire spectrum of anomalies that the system must face. This method must respond to the various constraints that weigh on the execution of such a system inside a vehicle. Vehicular networks are unique in that they operate services of two kinds. First, those related to the use of the vehicle in the context of intelligent transport systems (ITS) require that the vehicle regularly shares the interpretation of its environment as well as its state with a remote interlocutor. Indeed, modern vehicles are equipped with a set of sensors and actuators connected by an internal network. The information that this equipment generates is processed by electronic control units (ECU) to allow vehicles to make complex decisions such as detecting and avoiding obstacles. By sharing this knowledge with a centralization server, other vehicles can be warned of any dangers and adapt their driving. Similarly, monitoring the wear and tear of the equipment or the energy consumption of a fleet of vehicles is thus made possible by these telemetry services. The HTM algorithm has many properties making it suitable for use in the context of anomaly detection in vehicular networks.

1) Noise resistance: In the context of anomaly detection, noise can be thought of as small variations in attributes of communications in the VANET environment. For example,

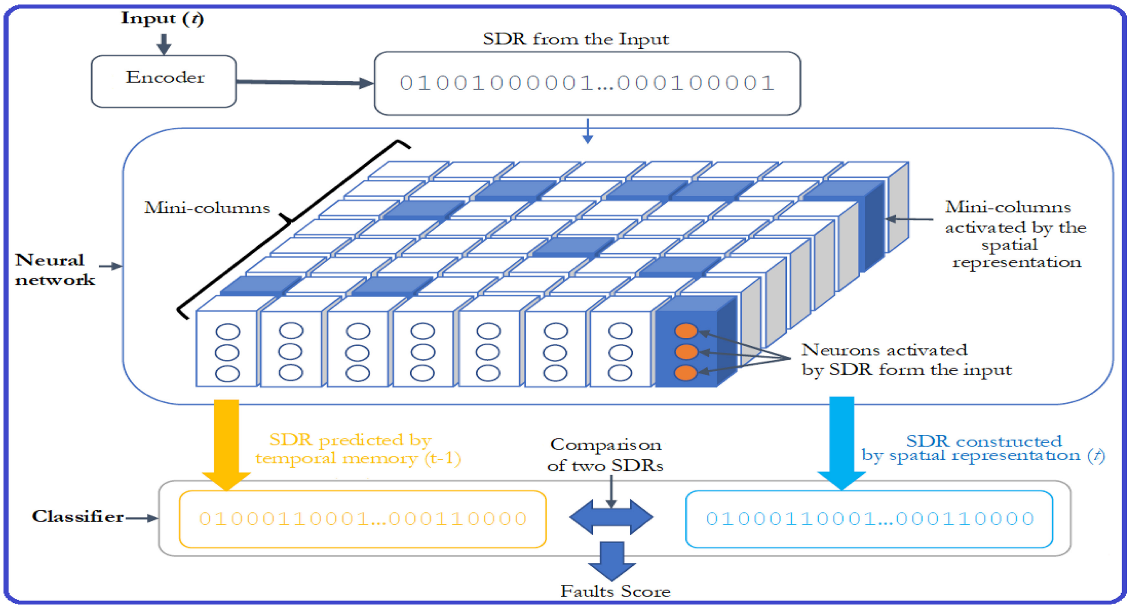
Input (t)
Encoder
SDR from the Input
01001000001...000100001
Mini-columns
Neural network
Mini-columns activated by the spatial representation
Neurons activated by SDR form the input
SDR predicted by temporal memory (t-1)
Comparison of two SDRs
SDR constructed by spatial representation (t)
Classifier
01000110001...000110000
01000110001...000110000
Faults Score

**FIGURE 2.** HTM Operation.

small variations in throughput do not necessarily mean that an attack is in progress or that a fault has occurred. The same is true for the loss of a few packets. All of this can simply be the consequence of benign events due to packet delay on the VANET network, network congestion, or the quality of the vehicle's cellular signal. It is therefore necessary for a good fault detector not to generate false positives because of these events. which, from a security point of view, do not have a significant impact. HTM learning is resistant due to the structure and organization of the neurons used by the algorithm.

2) Continuous learning: Unlike other methods like HMM [22] or LSTM [57], HTM continuously modifies the structure of the neural network by constantly manipulating the links between each neuron. These are reinforced or reduced according to the predictions made by the algorithm. This plasticity therefore also makes it capable of forgetting links if they are not often called upon. However, when it is considered that the traffic of the vehicle is subject to any change, this property is essential for the proper functioning of an anomaly detector.

3) Context sensitive predictions: The structure of neurons in HTM provides a memory to the algorithm that allows it to base its predictions on the current event as well as events it may have encountered in the past. The same entry can thus produce different results depending on the past context and the algorithm is theoretically able to recognize infrequent traffic without categorizing it as abnormal.

4) Each prediction detects faults: Each prediction made by the HTM algorithm indicates whether the current event was expected or not, which is crucial for fault detection. The algorithm may not be able to accurately predict the next event, but the memory modeling of the algorithm makes it able to predict multiple possible events at the next instant. Therefore, if he detects that an event is in no way what he expected, then it is abnormal. Moreover, unlike classical artificial neural network models, HTM does not need large amounts of data to obtain good classification results [55], [56].

### 2) OPERATION AND OVERVIEW OF HTM APPLIED TO OUR MODEL

HTM is a set of structures and algorithms, based on the following components (Fig. 2):

- The encoder of the data presented as input to the algorithm.
- A neural network composed of a set of mini-columns in which several pyramidal neurons reside.
- The spatial representation (or Spatial Pooler) and temporal memory algorithm, which manipulates the links between the neurons of the network according to the inputs of the algorithm.
- A classifier responsible for judging whether an input is abnormal or not.

The operation of the algorithm can be summarized as follows:

1) At time (*t*), the encoder converts the inputs presented to the algorithm into fixed-size binary vectors, also called Sparse Distributed Representations (SDRs).
2) The spatial representation algorithm, triggers from the SDR of the input, the activation of a small quantity of mini-columns of the neural network. The set of mini-columns of the network is represented in the form of another SDR, where the active bits represent the active mini-columns at time (*t*).

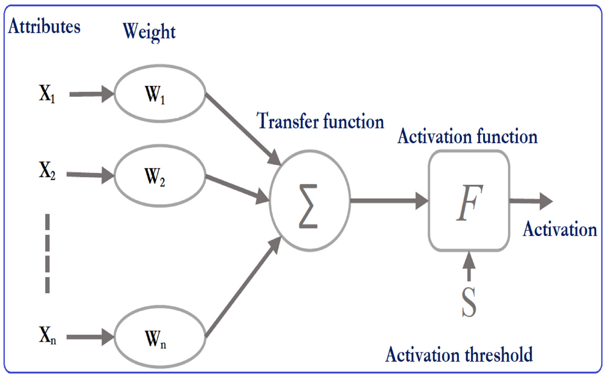

**FIGURE 3.** **Structure of the neuron commonly used in neural networks.**

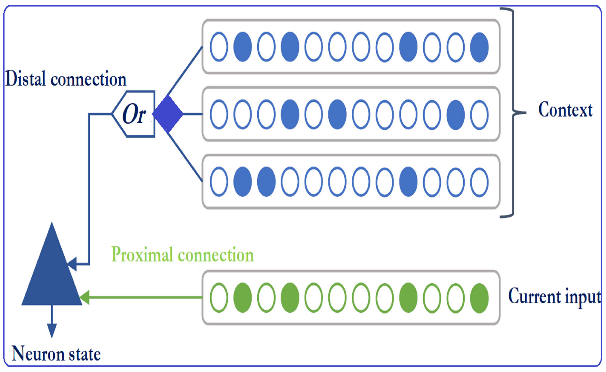

**FIGURE 4.** **Pyramidal structure of neurons used by HTM.**

3) The temporal memory algorithm, learns the sequences of mini-column activation and makes predictions about the future state of the network. It therefore produces an SDR representing the mini-columns whose activity at time (*t*) has been predicted at time (*t-1*).
4) These predictions are then presented to the classifier. In our case, we use a function that calculates the faults score of an input by comparing the SDR produced by the spatial representation at time (*t*) and the SDR predicted at time (*t-1*) by l temporal memory algorithm.

The spatial representation and temporal memory are therefore two algorithmic concepts that together govern the learning process of HTM by manipulating the links of the neurons of the network according to the SDRs presented by the encoder at the input of the network.

### 3) HTM LEARNING AND THE STRUCTURE OF THE NEURONS USED

Before detailing the key points of the functioning of HTM learning, we must first present the neuron model used by the HTM algorithm, because they differ from more widespread neural networks in particular by their pyramidal structure. Classical neural networks are composed of neurons whose output state (active or inactive) depends on: the weighted sum of the attributes presented as input, an activation function and the threshold from which the neuron is considered to be active or inactive (Fig. 3). The neurons used in HTM have two types of connections: proximal and distal (Fig. 4) and 3 possible output states: active by proximal connection, active by distal connection or inactive. These output states are directly related to the HTM network structure and the spatial representation and temporal memory algorithms. The advantage of the proximal connection is that each mini-column containing several neurons allows the algorithm to produce different predictions for the same input. More precisely, for mini-columns of x neurons and w mini-columns active per iteration, there are x w different ways to describe the same input in different contexts [5]. Similarly, the interest of the distal connection is that it serves as a predictive mechanism, where each time a neuron is activated by a proximal segment it has a chance to activate other neurons via the distal segments to which it is connected. If a neuron in a predictive state is then activated by a proximal segment at the next iteration, i.e., by an input presented to the network, it means that the prediction was good and that the distal connections must be reinforced. The reinforcement of these links will result in training the neurons of a mini-column to recognize a certain number of patterns and allow them to predict those that are supposed to appear in the following iterations.

### 4) EXAMPLE ON THE PREDICTIVE STATE AND LEARNING FOR FAULT DETECTION

In order to clarify the notion of predictive state and learning we describe a simplified example in the context of anomaly detection. To do this, we study the activation of the neurons of a simplified network as described by Fig. 3. In our example, the neural network is made up of 16 mini-columns of 4 neurons each. At the initialization of the algorithm, each flag presented for the first time to the network triggers the firing of each neuron of the mini-columns that have been activated by the different flags, which is illustrated by the first line of the figure. Now suppose that we submit a sufficient number of times this same sequence to the algorithm. The distal links between the neurons of a mini-column activated by a flag will be reinforced until, on seeing a flag, the neurons are able to predict the next flag. This behavior is illustrated by the second line. For example, when the SYN flag is received, then 3 mini-columns are activated (in blue) and 3 neurons switch to a predictive state (in orange). We then notice that when the next flag is received (SYN/ACK), the same neurons that were in the predictive state become active at this iteration while three others predict that the next flag will be ACK. The algorithm was therefore able to learn the SYN, SYN/ACK, ACK, PSH/ACK sequence which corresponds to the establishment of the TCP handshake and then the transmission of a data packet. If now we are interested in the third line of the figure where we submit another sequence to this same network, we notice that the prediction of the last flag (FIN/ACK) is erroneous. Indeed, the predicted flag was ACK and the neurons that were in predictive state (in red) are not those active when

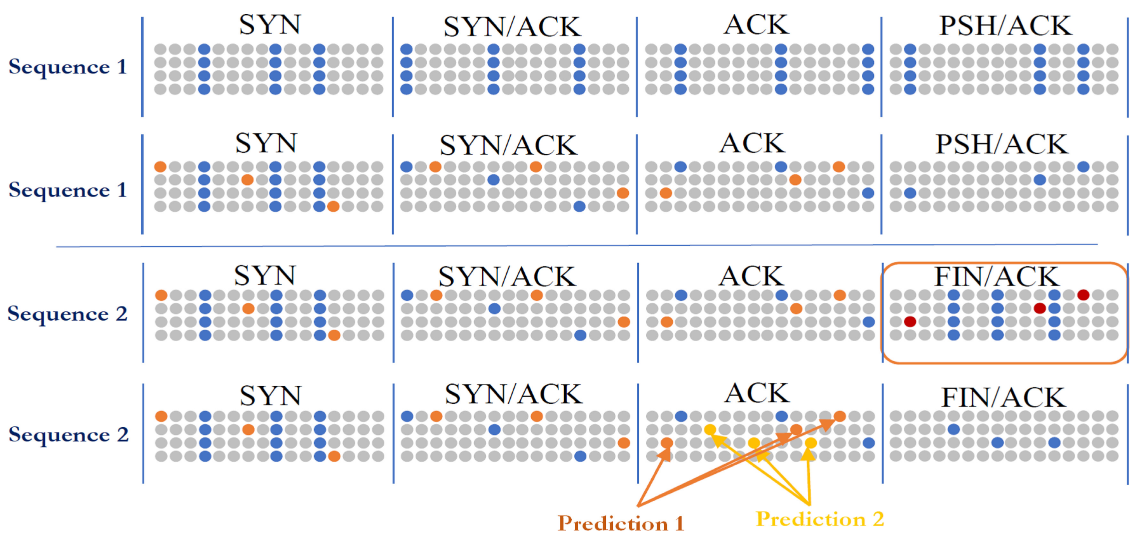

FIGURE 5. Illustration of HTM neuron learning and prediction process.

FIN/ACK is processed by the algorithm. Thus, the SYN, SYN/ACK, ACK, FIN/ACK sequence would be considered abnormal by the algorithm. However, if the sequence were to be presented a sufficient number of times to the algorithm, it would be able to get used to this new sequence. This behavior is illustrated by the last line where when the ACK is received, 2 predictions are then possible either the PSH/ACK of sequence 1, or the FIN/ACK of sequence 2. And when the FIN/ACK is finally received we finds that the neurons assets match the second prediction (in yellow). Consequently, the algorithm is then able to adapt continuously to the evolution of the traffic, thus limiting the probabilities that the algorithm becomes inefficient. This learning property is crucial because it allows the recognition of different sequences in the data observed by the algorithm, which makes it robust to changes in network behavior while being able to detect faults when they appear.

### C. MODEL ARCHITECTURE

In our model architecture, three layers can be identified: Edge layer, Fog Layer, and Cloud Layer.

- Edge layer (Things layer): The edge layer is where data is generated or gathered from the physical world. This layer involves the connected terminal devices, working to provide the system with gathered data. Each vehicle in this layer is having a communication model allowing the node to transmit the gathered or generated data to the upper layer (fog layer).
- Fog layer: involving, in each location, numerous decentralized nodes. This layer examines the primary measurement, refinement and processing of Edge-generated data. The main aim of fog nodes is to enhance the efficiency of IoT services; thus, the fog can decrease the amount of transmitted data to the upper layer (cloud layer) in addition to reduce response time for IoV services. In our context, this layer is used to discover and recover faulty measurements.
- Cloud layer: Cloud is the top layer of the IoV architecture, which enables network access to shared resources over the IoV network. Thus, Cloud performs the “heavy services” i.e., data saving and processing that fog is not able to perform, such as big data processing.

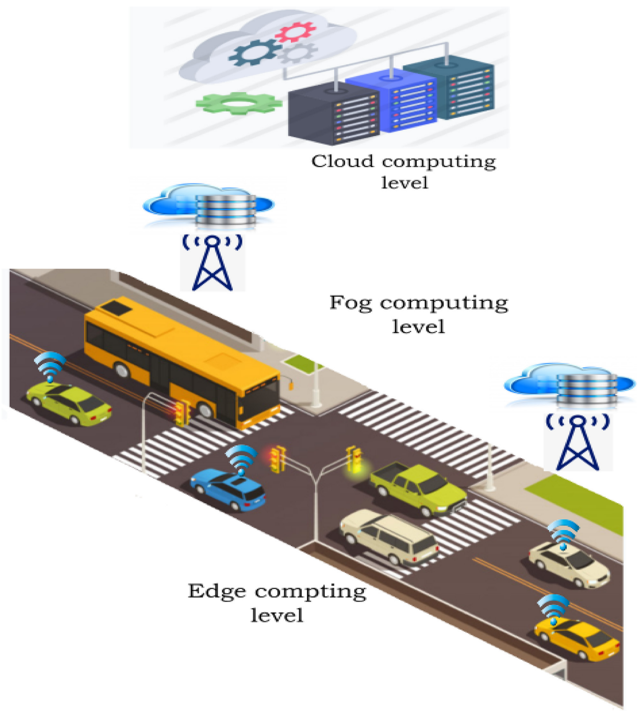

FIGURE 6. The model architecture contains three levels: The edge layer (Vehicles layer): is where data is generated or gathered. The fog involves numerous decentralized local server where that fault is detected and next recovered. The cloud layer of the IoV architecture enables network access to shared resources, and performs the “heavy services” of data processing.

TABLE 1. Features of the prepared dataset.

| Feature | Signification |
|---|---|
| Taxi Id | The id of Taxi |
| Latitude | The latitude position of the taxi |
| Longitude | The longitude position of the taxi |
| Temperature | The measured temperature |
| Temperature_1 | The measured temperature of neighbor 1 |
| Temperature_2 | The measured temperature of neighbor 2 |
| Temperature_3 | The measured temperature of neighbor 3 |
| Neighbor_1 | The Id of neighbor 1 |
| Neighbor_2 | The Id of neighbor 2 |
| Neighbor_3 | The Id of neighbor 3 |
| Class | Normal or the type of injected fault |

The model proposed in this article suggests that the fault detection is to be applied at the fog layer rather than the cloud layer. By doing so, the Quality of Service is enhanced as latency is reduced, and network bandwidth is improved.

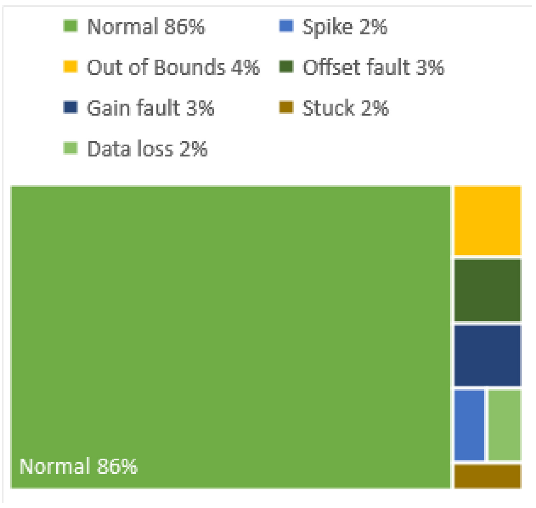

FIGURE 7. Data distribution in the dataset

## IV. EXPERIMENTAL STUDY

In this section, the details of the experiment are described, which is performed in four phases. During phase1, dataset is selected, in phase2, the faults are injected. Phase 3 consists in detecting faults using ML. Finally, in phase 4, the fault recovery process is done at fog level.

### A. DATASET SELECTION

The used data set is linked to a temperature measurement collected by taxi in Rome during February, 2014 [32]. The vector of data contains: the time stamp, the vehicle Id, the vehicle GPS location, and the weather temperature. This dataset has been used earlier in different research areas such as crowdsensing [33], data processing [34], and resource allocation [35]. However, and to the best of our knowledge, this study is the first study that makes use of this dataset in the context of fault detection research.

### B. DATASET MODIFICATION

As mentioned earlier, there are four attributes in the original dataset which are the vehicle Id, the time stamp, the GPS location, and the measured temperature. Two modifications have been applied to the original dataset. First, the dataset was injected with the six types of faults that we aim to detect. These include the following:

1) Data loss fault
2) Stuck-at fault
3) Spike fault
4) Out of bounds
5) Gain fault
6) Offset fault

The second modification that was applied to the original dataset was the addition of a set of temperature data with were gathered by the neighbors of each vehicles based on geographic distance. This will help later in the recovery process. The new dataset is described in Table 1. The dataset contains 4114 vectors, in the experiment 70% are used for training and the rest for testing. In Figure 7 we show the distribution of our outcomes.

The geographic distance is explained in the following subsection.

### C. GEOGRAPHIC DISTANCE

If $\varphi$ is the latitude, $\lambda$ is the longitude and $R$ is the earth mean radius ($6371km$), the geographic distance [30] between two vehicle, $v_1$ and $v_2$ with respectively the coordinates $(\varphi_1, \lambda_1)$ and $(\varphi_2, \lambda_2)$ can be calculated as follow:

$$a = hav(\Delta\varphi) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot hav(\Delta\lambda) \tag{8}$$

The haversine function is:

$$hav(\theta) = \sin^2\left(\frac{\theta}{2}\right) \tag{9}$$

Finally, the distance is:

$$dist(v_1, v_2) = 2 \cdot R \cdot \arctan\left(\sqrt{a}, \sqrt{1-a}\right). \tag{10}$$

### D. CLASSIFICATION METHODS' CONFIGURATIONS AND EVALUATION

This section presents the parameters' configurations the were used to train the four used methods. It also shows the evaluation metrics used.

#### 1) PARAMETERS' CONFIGURATIONS

The training dataset was splitted into two parts, one for training and the second for testing. The four models were trained on the training dataset and then tested to measure their performance.

#### 2) EVALUATION METRICS

The list of evaluation metrics used in this paper, are defined below. These well-known metrics are usually used to evaluate the results of classification techniques.

The precision metric is defined as follow by the equation (11):

$$precision = \frac{TP}{TP + FP} \tag{11}$$

The recall is defined as follow by the equation (12):

$$recall = \frac{TP}{TP + FN} \tag{12}$$

The $f1\text{-}score$ is a metric which combine the precision and the recall as follow by the equation (13):

$$f1\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \tag{13}$$

Finally, equation (14) define the accuracy which is the most important metric to evaluate the outcome of a classification:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{14}$$

where:

- $TP$: True positive: True fault which is detected correctly,
- $TN$: True negative: True fault which is not detected,
- $FP$: False positive: Correct value which is considered as fault,
- $FN$: False negative: Correct value which is not considered as fault.

### E. RECOVERY METHOD

The recovery of the correct measurement is based on mean value and the standard deviation [31]. Let $V$ the set of vehicles and $k$ the number of neighbors of the vehicle $v$ with faulty temperature in a period of time $\tau$, within a geographic distance that does not exceed $\delta$, so this set of neighbors vehicle can be represented by the equation (15):

$$N_1^\tau(v) = \{v_i \in V, dist(v, v_i) \preceq \delta \text{ and } |t - t_i| \preceq \tau\} \tag{15}$$

The average of all measured temperature can be calculated according to the following by the equation (16):

$$\bar{t} = \frac{\sum_{i=1}^{k} t_i(v_i)}{k} \tag{16}$$

A standard deviation $\sigma_t$ is calculated by the equation (17):

$$\sigma_t = \frac{1}{k}\sqrt{\sum_{i=1}^{k}\left(t_i - \bar{t}\right)^2} \tag{17}$$

All measured temperature exceeding $\sigma_t$ will be removed, i.e., if $|t_i - \bar{t}| > \sigma_t$. The number of the non-retrieved temperature is $k_{cor}$ (with $k_{cor} \leq k$), The recovered temperature will be calculated as follow (18):

$$\overline{t_{cor}} = \frac{\sum_{i=1}^{k_{cor}} t_i(v_i)}{k} \tag{18}$$

In the experiment we choose the three closed neighbors in the zone of $1km$ within the minimum time. Then we use the three collected temperature in the recovery process as explained previously. To measure the recovery rate the following equation is used (19):

$$difference\ percentage = \frac{|Real_{value} - Recov_{value}|}{Real_{value}}. \tag{19}$$

TABLE 2. Fault detection results using RF.

| RF Results | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-score** | **Support** |
| | | | | |
| Normal | 1.0000 | 1.0000 | 1.0000 | 6 |
| Offset fault | 0.6471 | 0.9167 | 0.7586 | 12 |
| Out of bounds | 0.9619 | 0.9888 | 0.9751 | 357 |
| Gain fault | 0.8750 | 0.5833 | 0.7000 | 12 |
| Data loss | 0.9000 | 0.6923 | 0.7826 | 13 |
| Spike | 0.0000 | 0.0000 | 0.0000 | 7 |
| Stuck | 0.3333 | 0.2000 | 0.2500 | 5 |
| | | | | |
| Accuracy | | | 0.9393 | 412 |
| Macro avg | 0.6739 | 0.6259 | 0.6381 | 412 |
| Weight avg | 0.9248 | 0.9393 | 0.9297 | 412 |
| | | | | |
| **Precision** | 92.48 | | | |
| **Accuracy** | 93.93 | | | |
| **Recall** | 93.93 | | | |

TABLE 3. Fault detection results using DT.

| DT Results | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-score** | **Support** |
| | | | | |
| Normal | 1.0000 | 1.0000 | 1.0000 | 6 |
| Offset fault | 0.5714 | 0.6667 | 0.6154 | 12 |
| Out of bounds | 0.9679 | 0.9300 | 0.9486 | 357 |
| Gain fault | 0.5385 | 0.5833 | 0.5600 | 12 |
| Data loss | 0.7333 | 0.8462 | 0.7857 | 13 |
| Spike | 0.0667 | 0.1429 | 0.0909 | 7 |
| Stuck | 0.1667 | 0.2000 | 0.1818 | 5 |
| | | | | |
| Accuracy | | | 0.8883 | 412 |
| Macro avg | 0.5778 | 0.6241 | 0.5975 | 412 |
| Weight avg | 0.9119 | 0.8883 | 0.8993 | 412 |
| | | | | |
| **Precision** | 91.19 | | | |
| **Accuracy** | 88.83 | | | |
| **Recall** | 88.83 | | | |

## V. RESULTS

In this section, the results of fault detection as well as recovery are presented. In this section, the result of fault detection based on the four classifier are outlined and discussed. Then, the recovery results are shown.

### A. FAULT DETECTION RESULTS

SVM, DT, RF and NN are runned using python on the dataset. Table 2 shows the results of using RF classifier. If we analyse by type of fault, we can see out of bounds fault and data loss fault are detected with good precision. However, other type like spike faults are not detected because of that values are very close to normal. The overall results will be compared next with other classifiers.

In Table 3, results of using DT are given. Best results are always with out of bounds fault with a precision of more than 96%. Spike and stuck fault are detected with a precision better than RF classifier but still not enough.

TABLE 4. Fault detection results using SVM.

| SVM Results | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-score** | **Support** |
| | | | | |
| Normal | 0.0000 | 0.0000 | 0.0000 | 6 |
| Offset fault | 0.0000 | 0.0000 | 0.0000 | 12 |
| Out of bounds | 0.8665 | 1.0000 | 0.9285 | 357 |
| Gain fault | 0.0000 | 0.0000 | 0.0000 | 12 |
| Data loss | 0.0000 | 0.0000 | 0.0000 | 13 |
| Spike | 0.0000 | 0.0000 | 0.0000 | 7 |
| Stuck | 0.0000 | 0.0000 | 0.0000 | 5 |
| | | | | |
| Accuracy | | | 0.8665 | 412 |
| Macro avg | 0.1238 | 0.1429 | 0.1326 | 412 |
| Weight avg | 0.7508 | 0.8665 | 0.8045 | 412 |
| | | | | |
| **Precision** | 75.08 | | | |
| **Accuracy** | 86.65 | | | |
| **Recall** | 86.65 | | | |

TABLE 5. Fault detection results using NN.

| Neural Network Results | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-score** | **Support** |
| | | | | |
| Normal | 1.0000 | 1.0000 | 1.0000 | 6 |
| Offset fault | 0.9097 | 0.8333 | 0.8696 | 12 |
| Out of bounds | 0.9545 | 1.0000 | 0.9767 | 357 |
| Gain fault | 1.0000 | 0.6667 | 0.8000 | 12 |
| Data loss | 0.8462 | 0.8462 | 0.8462 | 13 |
| Spike | 0.0000 | 0.0000 | 0.0000 | 7 |
| Stuck | 0.0000 | 0.0000 | 0.0000 | 5 |
| | | | | |
| Accuracy | | | 0.9515 | 412 |
| Macro avg | 0.6728 | 0.6209 | 0.6418 | 412 |
| Weight avg | 0.9240 | 0.9515 | 0.9362 | 412 |
| | | | | |
| **Precision** | 92.40 | | | |
| **Accuracy** | 95.15 | | | |
| **Recall** | 95.15 | | | |

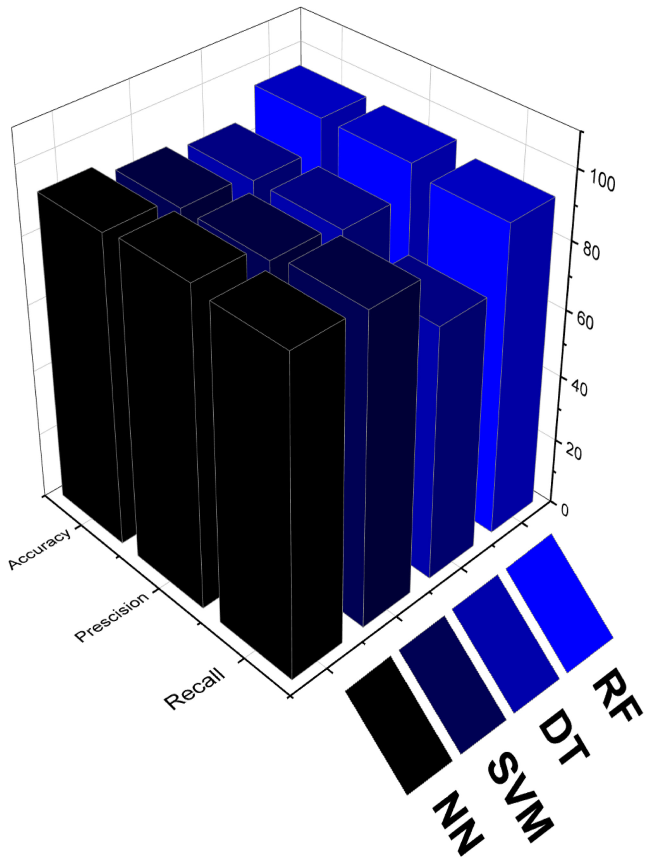

FIGURE 8. Overall fault detection results comparison: despite the fact that DT outperforms other methods based on fault type comparison. But, the overall comparison shows that NN outperforms other classifier in term of precision, recall and accuracy (Accurancy of NN is 95.15%, that of RF is 93.93%, 88.83% for DT and only 86.56% for SVM.).

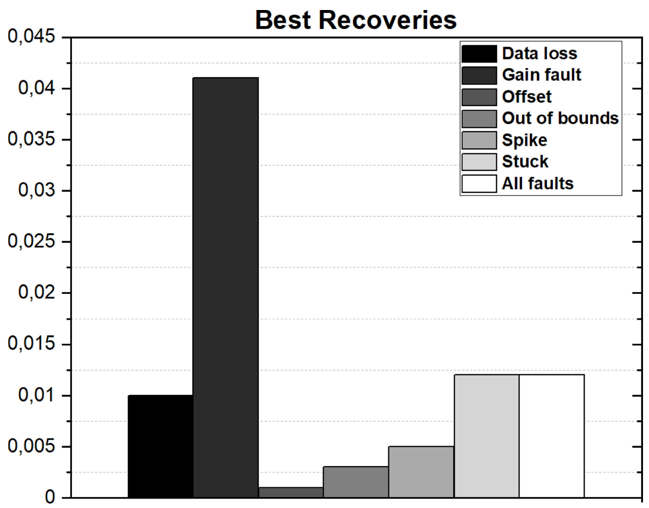

FIGURE 9. Best recovery representing the best percentage of difference between real and recovered value over real value. The best recovery rate is for offset fault and the worst one is for gain fault.

SVM results are presented in Table 4. By type of faults, SVM classifier gives the worst results. Since SVM can't detect most of faults. The only detected fault is out of bounds with acceptable values.

In Table 5, results of using NN are shown. We can see the NN performs well in detecting faults like gain fault. However, still problems with spike and stuck fault due to fact that these faults are close to normal values.

To make an overall comparison of the four classifier results, Fig. 8 shows the performance of RF, DT, SVM, and NN in terms of precision, accuracy and recall. Previous figures show that DT outperforms other methods based on fault type comparison. Since DT detects all type of faults even with low success. However, the overall comparison shows that NN outperforms other with an accuracy more than 95%, that of RF is 93.93%, 88.83% for DT and only 86.56% for SVM.

## B. FAULT RECOVERY RESULTS

Table 5 shows a comparison of the recovery results of the several types of faults in a scale of percentage. The values in the table represents the difference between real and recovered values according to the following equation:

Fig. 9, 10 and 11 are representing respectively, best, average and worst recoveries.

The best recovery is representing the best percentage of difference between real and recovered value over real value.

TABLE 6. Set up parameters and recovery results.

| | Loss | Gain | Offset | O of B | Spike | Stuck | All |
|---|---|---|---|---|---|---|---|
| **Worst** | 0.991 | 0.845 | 0.738 | 0.814 | 0.973 | 0.960 | 0.887 |
| **Best** | 0.010 | 0.041 | 0.001 | 0.003 | 0.005 | 0.012 | 0.012 |
| **Average** | 0.199 | 0.648 | 0.403 | 0.576 | 0.204 | 0.210 | 0.373 |
| **Distance** | $\delta = 1000m$ | | | | | | |
| **N** | number of neighbors = 3 | | | | | | |
| **Time** | $\tau$ = 60 seconds | | | | | | |

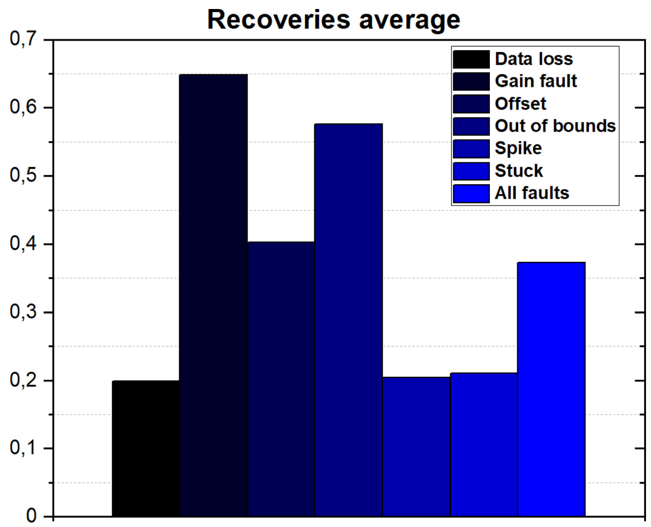

FIGURE 10. Average recovery representing the average percentage of difference between real and recovered value over real value. The best average recovery rate is for data loss fault and the worst one is for gain fault.

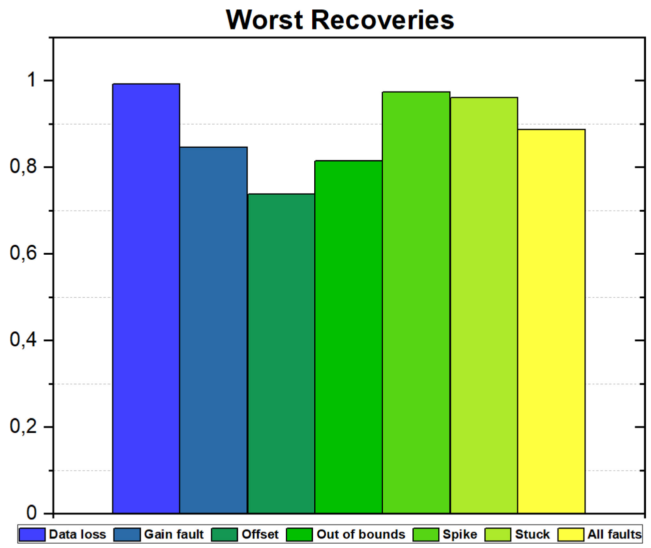

FIGURE 11. Worst recovery representing the worst percentage of difference between real and recovered value over real value. The worst recovery rate is for data loss fault and the less worst one is for offset fault.

According to Fig. 9, the best recovery rate is for offset fault by a percentage of difference of 0.001 for recovered value over real value. While, the worst one is for gain fault by a percentage of difference of 0.041 for recovered value over real.

The average recovery is representing the average percentage of difference between real and recovered value over real value. According to Fig. 10, the best average recovery rate is for data loss fault by a percentage of difference of 0.199 for recovered value over real value. While, the worst one is for gain fault by a percentage of difference of 0.648 for recovered value over real value.

The worst recovery is representing the worst percentage of difference between real and recovered value over real value. According to Fig. 11, the worst recovery rate is for data loss fault by a percentage of difference of 0.991 for recovered value over real value. While, the less worst one is for offset fault by a percentage of difference of 0.738 for recovered value over real value. The recovery rate is depending on the number of neighbors and the density of vehicles. In general, the more there neighbors and highest vehicle density the more the recovery rate is maximized.

## VI. CONCLUSION

In VoT, the exchange of message or the collect of data are very important. These messages are prone to many failures. In this paper, ML techniques are used to detect if exchanged data is faulty or not. The process of fault detection and recovery is done at fog level rather than cloud level. This fact allows to enhance the Quality of Service as latency is reduced, and network bandwidth is improved.

The fault detection and recovery is performed into two phases. At first stage, four classification methods were used which are SVM, DT, RF and NN. These techniques are applied in a well known dataset where 6 kinds of faults are injected. The detection based on type of fault shows that DT outperforms other classifier. However, an overall comparison shows that NN gives best results. The second stage consists in recovering the faulty data by aggregating data coming from neighborhoods.

We have also proposed the use of the Hierarchical Temporal Memory (HTM) algorithm which has shown its ability to adapt continuously to the evolution of VANET traffic, thus limiting the probability that the algorithm becomes inefficient. The learning property of HTM is crucial, as it allows the recognition of different sequences in the data observed by the algorithm, which makes it robust to changes in network behavior while being able to detect faults when they appear.

A shortcoming improvement of this work can be performed by considering malicious intrusion. Indeed IoV is an open network so that malicious node can easily be part of the network. Which make the fact of preventing and detecting attacks as important.

The scope of our study was to adopt only 6 faults (out of bounds, spike, stuck, offset, gain fault, and data loss) on the famous Rome taxis dataset. We used only four ML techniques (DT, RF, NN and SVM). We believe that our work can be generalized and enhanced by being applied to other bigger datasets using also other ML techniques, and with other faults like the random fault which can happen unexpectedly.

## REFERENCES

[1] J. Voelcker. "It's official: We now have one billion vehicles on the planet." 2011. Accessed: Dec. 28, 2019. [Online]. Available: https://www.greencarreports.com/news/1065070itsofficialwenowhaveonebillion-vehicles-on-the-planet

[2] A. Mchergui, T. Moulahi, B. Alaya, and S. Nasri, "A survey and comparative study of QoS aware broadcasting techniques in VANET," *Telecommun. Syst.*, vol. 66, no. 2, pp. 253–281, Oct. 2017.

[3] A. Mchergui, T. Moulahi, and S. Zeadally, "Survey on artificial intelligence (AI) techniques for vehicular ad-hoc networks (VANETs)," *Veh. Commun.*, vol. 34, Apr. 2022, Art. no. 100403.

[4] R. G. Engoulou, M. Bellaïche, S. Pierre, and A. Quintero, "VANET security surveys," *Comput. Commun.*, vol. 44, pp. 1–13, May 2014. [Online]. Available: https://doi.org/10.1016/j.comcom.2014.02.020

[5] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101823, doi: 10.1016/j.adhoc.2018.12.006.

[6] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, 2014, "CRAWDAD dataset roma/taxi (v.2014-07-17)," traceset: taxicabs. [Online]. Available: https://crawdad.org/roma/taxi/20140717/taxicabs

[7] A. M. Malla and R. K. Sahu, "Security attacks with an effective solution for DOS attacks in VANET," in *Proc. Int. J. Comput. Appl.*, vol. 66, 2013, pp. 45–49.

[8] S. S. Manvi and S. Tangade, "A survey on authentication schemes in VANETs for secured communication," *Veh. Commun.*, vol. 9, pp. 19–30, Jul. 2017. [Online]. Available: https://doi.org/10.1016/j.vehcom.2017.02.001

[9] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 78, pp. 267–287, Jan. 2017. [Online]. Available: https://doi.org/10.1016/j.jnca.2016.10.019

[10] P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 30, pp. 291–319, Jul. 2018. [Online]. Available: https://doi.org/10.1016/j.jksuci.2016.10.003

[11] S. Sharma and B. Kaushik, "A survey on Internet of Vehicles: Applications, security issues & solutions," *Veh. Commun.*, vol. 20, Dec. 2019, Art. no. 100182. [Online]. Available: https://doi.org/10.1016/j.vehcom.2019.100182

[12] B. Alaya, "Payoff-based dynamic segment replication and graph classification method with attribute vectors adapted to urban VANET," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 17, no. 3, pp. 1–22, 2021. [Online]. Available: https://doi.org/10.1145/3440018.

[13] F. Yang, J. Li, T. Lei, and S. Wang, "Architecture and key technologies for Internet of Vehicles: A survey," *J. Commun. Inf. Netw.*, vol. 2, no. 2, pp. 1–17, 2017. [Online]. Available: https://doi.org/10.1007/s41650-017-0018-6

[14] P. M. Khilar and S. Mahapatra, "Intermittent fault diagnosis in wireless sensor networks," in *Proc. 10th Int. Conf. Inf. Technol. (ICIT)*, Apr. 2008, pp. 145–147, doi: 10.1109/icit.2007.15.

[15] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-based mechanisms to discriminate transient from intermittent faults," *IEEE Trans. Comput.*, vol. 49, no. 3, pp. 230–245, Mar. 2000, doi: 10.1109/12.841127.

[16] M. Panda, B. S. Gouda, and T. Panigrahi, "Fault diagnosis in wireless sensor networks using a neural network constructed by deep learning technique," in *Nature Inspired Computing for Wireless Sensor Networks* (Springer Tracts in Nature-Inspired Computing). Singapore: Springer, 2020, pp. 77–101.

[17] R. R. Panda, B. S. Gouda, and T. Panigrahi, "Efficient fault node detection algorithm for wireless sensor networks," in *Proc. Int. Conf. High Perform. Comput. Appl.*, Feb. 2015, pp. 1–5, doi: 10.1109/ICHPCA.2014.7045308.

[18] K. Hornik, M. Stinchcombe, and H. White, "'Multilayer feedforward networks are universal approximators,'" cs.cmu.edu. Accessed: Mar. 2, 2020. [Online]. Available: https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2016/notes/Sonia_Hornik.pdf

[19] M. Panda and P. M. Khilar, "Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test," *Ad Hoc Netw.*, vol. 25, pp. 170–184, Feb. 2015, doi: 10.1016/j.adhoc.2014.10.006.

[20] M. Saihi, B. Boussaid, A. Zouinkhi, and M. N. Abdelkrim, "Decentralized fault detection in wireless sensor network based on function error," in *Proc. 10th Int. Multi-Conf. Syst., Signals Devices*, 2013, pp. 1–5, doi: 10.1109/SSD.2013.6564159.

[21] H. Yuan, X. Zhao, and L. Yu, "A distributed Bayesian algorithm for data fault detection in wireless sensor networks," in *Proc. Int. Conf. Inf. Netw.*, Mar. 2015, pp. 63–68, doi: 10.1109/ICOIN.2015.7057858.

[22] E. U. Warriach and K. Tei, "Fault detection in wireless sensor networks: A machine learning approach," in *Proc. IEEE 16th Int. Conf. Comput. Sci. Eng.*, Dec. 2013, pp. 758–765, doi: 10.1109/CSE.2013.116.

[23] A. Karmarkar, P. Chanak, and N. Kumar, "An optimized SVM based fault diagnosis scheme for wireless sensor networks," in *Proc. IEEE Int. Students' Conf. Electr., Electron. Comput. Sci. (SCEECS)*, 2020, pp. 1–7. [Online]. Available: https://ieeexplore-ieee-org.sdl.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=9087027

[24] R. R. Swain and P. M. Khilar, "composite fault diagnosis in wireless sensor networks using neural networks," *Wireless Pers. Commun.*, vol. 95, no. 3, pp. 2507–2548, Aug. 2017, doi: 10.1007/s11277-016-3931-3.

[25] B. C. P. Lau, E. W. M. Ma, and T. W. S. Chow, "Probabilistic fault detector for wireless sensor network," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3703–3711, Jun. 2014, doi: 10.1016/j.eswa.2013.11.034.

[26] C. Titouna, M. Aliouat, and M. Gueroui, "Outlier detection approach using bayes classifiers in wireless sensor networks," *Wireless Pers. Commun.*, vol. 85, no. 3, pp. 1009–1023, 2015, doi: 10.1007/s11277-015-2822-3.

[27] M. S. Yadav and S. Ahamad, "Outlier detection in wireless sensor networks data by entropy based K-NN predictor," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, pp. 5483–5489, Oct. 2019, doi: 10.35940/ijitee.K2290.1081219.

[28] H. Zhang, "The optimality of Naïve Bayes," in *Proc. FLAIRS Conf.*, 2004, pp. 1–16. Accessed: Jun. 15, 2020. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.483.2183

[29] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sensors J.*, vol. 18, no. 1, pp. 340–347, Jan. 2018.

[30] "Calculate distance between GPS points in Python." Feb. 24, 2020. [Online]. Available: https://janakiev.com/blog/gps-points-distance-python/

[31] M. Lehsaini, M. Feham, and H. Guyennet, "Efficient cluster-based fault-tolerant schemes for wireless sensor networks," in *Proc. 5th Int. Conf. New Technol., Mobility Security (NTMS)*, 2012, pp. 1–5.

[32] R. Amici, M. Bonola, L. Bracciale, A. Rabuffi, P. Loreti, and G. Bianchi, "Performance assessment of an epidemic protocol in VANET using real traces," in *Proc. MoWNet*, Dec. 2014, pp. 92–99.

[33] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang, "An efficient prediction-based user recruitment for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 16–28, May 2018, doi: 10.1109/tmc.2017.2702613.

[34] C. Chilipirea, A. C. Petre, L. M. Groza, C. Dobre, and F. Pop, "An integrated architecture for future studies in data processing for smart cities," *Microprocess. Microsyst.*, vol. 52, pp. 335–342, Jul. 2017, doi: 10.1016/j.micpro.2017.03.004.

[35] L. Wang, L. Jiao, J. Li, and M. Muhlhauser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. Int. Conf. Distrib. Comput. Syst.*, Jul. 2017, pp. 1281–1290, doi: 10.1109/ICDCS.2017.30.

[36] T.-K. Dao, T.-T. Nguyen, J.-S. Pan, Y. Qiao, and Q.-A. Lai, "Identification failure data for cluster heads aggregation in WSN based on improving classification of SVM," *IEEE Access*, vol. 8, pp. 61070–61084, 2020.

[37] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A practical security architecture for in-vehicle CAN-FD," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2248–2261, Aug. 2016.

[38] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[39] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[40] U. D. Gandhi and R. V. S. M. Keerthana, "Request response detection algorithm for detecting DoS attack in VANET," in *Proc. Int. Conf. Rel. Optim. Inf. Technol.*, 2014, pp. 192–194.

[41] S. Roselinmary, M. Maheshwari, and M. Thamaraiselvan, "Early detection of DOS attacks in VANET using attacked packet detection algorithm (APDA)," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2013, pp. 237–240.

[42] M. C. Herrera, J. B. Almeida, and C. Iza, "Performance evaluation in misbehaviour detection techniques for DoS attacks in VANETs," in *Proc. 18th ACM Symp. Perform. Eval. Wireless Ad Hoc, Sens., Ubiquitous Netw.*, New York, NY, USA, 2021, pp. 73–80.

[43] M. N. Mejri, J. B. Othman, and M. Hamdi, "Survey on VANET security challenges and possible cryptographie solutions," *Veh. Commun.*, vol. 1, no. 2, pp. 53–66, 2014.

[44] A. M. Malla and I. V. K. Sahu, "Security attacks with an effective solution for DOS attacks in VANET," *Int. J. Comput. Appl.*, vol. 66, no. 22, pp. 45–49, Mar. 2013.

[45] L. He and W. T. Zhu, "Mitigating DOS attacks against signature-based authentication in VANETs," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng. (CSAE)*, 2012, pp. 261–265.

[46] K. Laroussi, A. B. Boucif, M. Mesfioui, and I. Biskri, "A probabilistic model to corroborate three attacks in vehicular ad hoc networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 70–75.

[47] J. Hawkins and S. Blakeslee, *On Intelligence: How a New Understanding of the Brain Will Lead to the Creation of Truly Intelligent Machines*. New York, NY, USA: Macmillan, 2007.

[48] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, 2016.

[49] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.

[50] Z. Hasani, "Robust anomaly detection algorithms for real-time big data: Comparison of algorithms," in *Proc. 6th Mediterr. Conf. Embedded Comput. (MECO)*, 2017, pp. 1–6.

[51] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu, and X. Cheng, "A distributed anomaly detection system for in-vehicle network using HTM," *IEEE Access*, vol. 6, pp. 9091–9098, 2018.

[52] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient KNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, Jun. 2016.

[53] N. Kouiroukidis, and G. Evangelidis, "The effects of dimensionality curse in high dimensional KNN search," in *Proc. 15th Panhellenic Conf. Informat.*, 2011, pp. 41–45.

[54] G. Fernandes Jr., L. F. Carvalho, J. J. P. C. Rodrigues, and M. L. Proença Jr., "Network anomaly detection using IP flows with principal component analysis and ant colony optimization," *J. Netw. Comput. Appl.*, vol. 64, pp. 1–11, Apr. 2016.

[55] J. Wu, W. Zeng, and F. Yan, "Hierarchical temporal memory method for timeseries-based anomaly detection," *Neurocomputing*, vol. 273, pp. 535–546, Jan. 2018.

[56] Y. Cui, S. Ahmad, and J. Hawkins, "The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding," *Front. Comput. Neurosci.*, vol. 11, p. 111, Nov. 2017.

[57] A. R. Abdellah and A. Koucheryavy, "VANET traffic prediction using LSTM with deep neural network learning," in *Proc. 20th Int. Conf. Internet Things, Smart Spaces, Next Gener. Netw. Syst.*, Aug. 2020, pp. 281–294. [Online]. Available: https://doi.org/10.1007/978-3-030-65726-0_25

[58] F. Qu, Q. Jiang, G. Jin, Y. Wei, and Z. Zhang, "Mud pulse signal demodulation based on support vector machines and particle swarm optimization," *J. Petrol. Sci. Eng.*, vol. 193, Oct. 2020, Art. no. 107432.