network

MDPI

Article

Machine Learning Framework for Detecting False Alerts in Safety Messages †

Avinash Karhana, Ikjot Saini and Arunita Jaekel *

School of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada; karhanaa@uwindsor.ca (A.K.); ikjot.saini@uwindsor.ca (I.S.)

* Correspondence: arunita@uwindsor.ca

† This paper is an extended version of our paper published in Karhana, A.; Saini, I.; Jaekel, A. Machine Learning Approach to False Alert Attack Detection in Vehicular Adhoc Networks. In Proceedings of the 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2025; pp. 1–7. http://doi.org/10.1109/CCNC54725.2025.10975893.

Abstract

The recent advances in Vehicular Ad Hoc Networks (VANETs) can have a tremendous positive impact on vehicle safety and traffic flow. In VANETs, vehicles communicate wirelessly with each other and with roadside infrastructure nodes to improve awareness of neighboring vehicles and traffic conditions. However, such communication also increases the potential for various safety and security challenges, such as the threat of false reporting attacks. In these attacks, malicious or compromised nodes inject alert notifications that report fictitious traffic incidents that may trigger unnecessary evasive actions and increase the risk of collisions. This research addresses false alert attacks in VANETs by developing a machine learning-based detection framework that leverages innovative feature engineering and model assessment strategies. The proposed framework designs new features that capture vehicle kinematics to improve the detection of malicious alerts. The performance of multiple ML models is then analyzed in terms of both detection effectiveness and computational requirements to determine their suitability for different deployment scenarios. Our simulation results demonstrate that the proposed framework can achieve significant improvements compared to existing techniques for false alert detection. In addition, the study highlights the critical role of feature engineering in improving detection performance.

Keywords: VANETs; VeReMiAP dataset; false alert attack; misbehavior detection system (MDS)

Check for updates

Academic Editor: Pingyi Fan

Received: 11 May 2026
Revised: 22 June 2026
Accepted: 9 July 2026
Published: 14 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Vehicular Ad Hoc Networks (VANETs) serve as a fundamental component of modern Intelligent Transportation Systems (ITSs), enabling communication among vehicles and between vehicles and roadside infrastructure. VANETs can help to enhance road safety, mitigate traffic congestion, and improve overall traffic management efficiency [1]. With the increasing complexity of transportation infrastructures, VANETs have become essential for supporting the efficient operation of road networks through the real-time exchange of critical information, including vehicle location, speed, and roadway conditions. Despite their many benefits, VANETs, like other interconnected systems, remain vulnerable to a variety of security threats that can undermine the reliability, integrity, and safety of transportation services [2].

Basic Safety Messages (BSMs) and Cooperative Awareness Messages (CAMs) are foundational to vehicular communication systems, enabling real-time exchange of position, speed, heading, and other motion parameters between nearby vehicles and infrastructure. BSMs are defined under SAE J2735 (U.S.) [3], while CAMs are standardized under ETSI EN 302 637-2 (Europe) [4]. CAMs emphasize cooperative awareness and environmental context sharing, supporting not just safety but also traffic efficiency and cooperative driving functions. BSMs, on the other hand, are more safety-centric, designed for direct collision avoidance and warning systems. The effectiveness of these applications depends on the trustworthiness of shared data. However, malicious or faulty nodes can transmit false or inconsistent information, leading to incorrect decisions and potential safety risks. Misbehavior detection is therefore essential to identify and isolate such nodes, ensuring the integrity, reliability, and safety of vehicular communication beyond what cryptographic authentication alone can provide. For example, a malicious vehicle may broadcast false position or speed information, deceiving nearby vehicles into making unsafe maneuvers such as sudden braking or lane changes. Similarly, in a Sybil attack, an adversary may create multiple fake identities to manipulate traffic flow or disrupt cooperative applications, such as platooning. Such misbehaviors undermine the reliability of safety messages, emphasizing the need for effective detection mechanisms to maintain trust and situational integrity within VANETs.

The security of VANETs is becoming a critical concern, with the increasing deployment of semi-autonomous and fully autonomous vehicles. False Alert Attack is a particularly serious threat targeting VANETs. In such attacks malicious nodes inject falsified accident reports or hazard warnings into the network to mislead neighboring vehicles and roadside infrastructure. There may be different variants of "false alert attacks", e.g., false notifications regarding road conditions, emergency vehicles approaching, accidents or other hazardous conditions. In this paper, we have limited our investigation specifically to the case where a vehicle falsely reports observing an accident at a location when there is no actual accident that occurred. Such deceptive messages may lead to traffic disruptions, misdirected vehicles, and various safety hazards [5]. Therefore, the development of effective detection and mitigation strategies is vital for ensuring the secure and dependable functioning of VANETs.

To address this challenge, this research proposes a machine learning (ML) approach for detecting false alert attacks in VANETs. Unlike conventional trust-based methods that rely on predefined metrics, the proposed approach leverages pattern recognition learned directly from vehicular communication data. Using the VeReMiAP dataset [6], which includes false alert messages in addition to legitimate broadcasts from vehicles, the model is trained to accurately distinguish between normal and malicious behavior. The important contributions of this work are:

- We implement an ML-based detection framework specifically designed for false alerts in VANETs, providing a systematic baseline for this emerging problem.
- We propose new engineered features that capture suspicious vehicular behaviors and kinematic inconsistencies to augment the original dataset representation and improve detection capability.
- We evaluate multiple ML models in terms of both predictive performance and computational cost to determine their suitability for different deployment settings.

The remainder of this paper is organized as follows. Section 2 first gives an overview of ML-based misbehavior detection in VANETs and then focuses on false alerts and the VeReMiAP dataset. Section 3 presents details of the proposed framework for false alert detection, and Section 4 discusses the performance of the proposed models. Finally, Section 5 presents the conclusion and some directions for future work.

https://doi.org/10.3390/network6030053

2. Related Work

Secure VANET communication, including misbehavior detection techniques for various types of attacks [7], as well as prevention and mitigation strategies [8], is an active area of research. Existing research on VANET security has proposed numerous attack detection mechanisms targeting a wide range of threats [9]. Machine learning (ML) has emerged as a promising approach for misbehavior detection by enabling the automated analysis of network data and the identification of anomalous or malicious behaviors [10]. Consequently, ML-based detection frameworks have been extensively investigated for various attacks, such as routing attacks, GPS spoofing, impersonation attacks, and DoS attacks. But the application of ML techniques for the detection of false alert attacks remains relatively underexplored, highlighting an important research gap in the current literature.

2.1. Machine Learning Based Misbehavior Detection in VANETs

In recent years, ML has been used successfully in a wide range of fields, such as healthcare, education, law and wireless communication [11–15]. Machine learning (ML) approaches are inherently data-driven, relying on representative datasets from which models can learn underlying patterns to make reliable predictions. However, due to security, privacy and safety concerns, access to real-world vehicular datasets remains limited. In 2018, Heijden et al. introduced a standardized, publicly extensible dataset, called VeReMi (Vehicular Reference Misbehaviour Dataset), based on the Luxembourg traffic scenario [16]. This dataset aims to facilitate fair and meaningful comparisons of different misbehavior detection approaches [17] and has been widely used to train and evaluate ML models as well as non-ML approaches. The VeReMi dataset focused on position falsification attacks and was extended, using F2MD [18] to generate the VeReMi Extension (VeReMiExt) dataset [19]. VeReMiExt introduced many new attacks, including DoS attacks, data replay, DoS disruptive attacks, traffic congestion Sybil attacks etc. In addition, many researchers have also generated and used their own datasets for performance evaluation.

In [20], the authors use ensemble learning to detect different routing attacks. In [21], integrity and plausibility checks are used in conjunction with ML to improve detection for false position attacks. Singh et al. [22] generated wormhole attacks using SUMO traffic simulator and NS3 and used KNN and SVM algorithms to detect these attacks. Gyawali et al. [23] applied various supervised ML algorithms including KNN, decision tree, bagging and random forest for false position attacks, with bagging classifiers achieving the best results. Sharma et al. [24] also used the VeReMi dataset to implement position falsification detection models. They used classical ML algorithms, reported that ensemble learning led improved predictions compared to individual models. Shams et al. [25] used a privately generated dataset to develop a trust-aware SVM model that improved packet delivery ratio and reduced end-to-end delay. In [26], the authors used unsupervised learning to detect jamming attacks in VANET. In addition to classical ML algorithms, a number of researches have investigated the use of deep learning and federated learning techniques for VANET IDS. In [27], a Convolution Neural Network (CNN) is used to detect anomalies in BSM data, using the Wyoming dataset [28]. Aneja et al. [29] used genetic algorithm-based feature selection, in conjunction with an Artificial Neural Network (ANN) to detect flooding attacks. Kamel et al. [30] presented a Recurrent Neural Network (RNN) and LSTM model to detect Sybil attacks. Zeng et al. [31] proposed an LSTM-based classification model capable of detecting different types of attacks, including DoS, routing and Sybil attacks.

2.2. Detecting False Alert Attacks

In VANETs, ensuring message authenticity is essential because false traffic warnings can compromise road safety. Liu et al. [32] proposed a traffic-flow-based verification

scheme that uses neighboring vehicles as witnesses to distinguish genuine emergency messages from false reports. Similarly, Ghaleb et al. [33] introduced a context-aware framework that combines unsupervised learning and Bayesian hypothesis testing to identify spurious messages and reduce false alarms. Although Liu et al. [32] and Ghaleb et al. [33] addressed false alert attacks, neither employed machine learning models specifically for attack detection. This may be due to the limited availability of datasets containing realistic false alert attack scenarios. In [34], a traffic anomaly index, calculated using a Gradient Boosting Decision Tree (GBDT) model, is used to differentiate between legitimate and malicious vehicles.

The Framework for Misbehavior Detection in Vehicular Networks [18] was further modified and extended by Abdelmaguid et al. to create the VeReMiAP dataset [35]. This dataset introduces false alert attacks in BSMs and CAMs and is used in this research to train and test our models. The authors also studied the effect of false alert attacks on neighboring vehicles to determine the potential for hazardous conditions due to false alerts. In addition to simulated datasets, the availability of real-world data would be invaluable for evaluating false alert detection methods, but no real-world dataset for such attacks currently exists. Although some real-world BSM data are available from various testbeds across the USA, they do not contain any false/malicious BSMs, and therefore are not directly useful for misbehavior detection. Therefore, at this time, detection approaches have to typically rely on simulated datasets.

VeReMiAP Dataset

The VeReMiAP (VeReMi for Attack prediction) dataset [6] provides a diverse range of data mimicking real-world driving scenarios, with vehicles communicating via BSMs [36] and CAMs [37]. These messages include critical information on vehicle dynamics enriched with 'noise' parameters to represent data uncertainties. BSMs convey vital information about vehicle dynamics, including speed, location, acceleration and other relevant parameters. In actual BSMs, the vehicle positions are expressed using three-dimensional coordinates (x, y, z). However, the VeReMiAP dataset is generated based on a two-dimensional simulation scenario. Therefore, the z-coordinates of the vehicle positions (as well as other three-dimensional parameters) are always set to 0 for all BSMs and CAMs in the dataset.

In addition to BSMs, the VeReMiAP dataset includes CAMs, which build upon the BSM framework by incorporating critical event-related data. CAMs are crucial for detailing situational awareness around vehicular environments, such as accidents and road conditions, providing a richer context with additional parameters like 'eventType', 'eventID', 'laneIndex', and 'lanePosition'. These details are essential for understanding and analyzing vehicular interactions under more complex scenarios within VANETs.

CAMs contain the original BSM fields, plus additional fields that provide a richer context about the ego vehicle as well as information about observed events. Table 1 shows the relevant BSM fields and additional CAM fields in the VeReMiAP dataset.

In this work, we focus on detecting false alert attacks within this dataset. Specifically, an attack is identified when a CAM inaccurately reports an Accident in the field of eventType in Table 1. without any corresponding incident near the reporting vehicle. Some preliminary results of our work in this area were reported in [38].

Table 1. BSM and CAM fields.

Field	Description
	Original BSM Fields
type	type of message BSM (=2) or CAM (=3)
rcvTime	time at which message was received

Table 1. *Cont.*

Field	Description
vehicleID	a unique identifier for a vehicle
pos	a composite value (pos_x, pos_y, pos_z) denoting the x, y, and z coordinates of a vehicle's position
pos_noise	a noise parameter related to position
spd	x, y, and z components of a vehicle's speed
spd_noise	a noise parameter related to speed
acl	x, y, and z components of a vehicle's acceleration
acl_noise	a noise parameter related to acceleration
hed	x, y, and z components of a vehicle's heading
hed_noise	a noise parameter related to heading
sender GPS	GPS coordinates of the message sender
sendTime	time a message is sent
sender	vehicle ID of the message sender
senderPseudo	a pseudonymous identifier for the sender
messageID	ID of the message
exchangedMessageType	type of message being exchanged, either a BSM or a CAM
	Additional CAM Fields
eventID	an identifier for a hazard event
eventType	the type of event that occurred
roadID	this corresponds to the "Road number" parameter
laneIndex	the index of the lane the vehicle is in
lanePosition	the position of the vehicle within the lane
maxSpeed	the maximum speed limit on the road
maxDeceleration	the maximum deceleration parameter
currentDirection	the current direction of the vehicle
hazardAttack	an additional field related to the "Fake Alert Attack" and its effect, which is a hazard
hazardOccurrence	a parameter related to the occurrence of a hazard
hazardOccurrencePercentage	a parameter related to the percentage of hazard occurrence

3. Proposed False Alert Detection Framework

In the realm of VANET, the threat of false alerts represents a critical challenge. These networks, crucial for improving road safety and traffic efficiency, rely on the accuracy and authenticity of exchanged messages. False alerts or deceptive messages about non-existent events like traffic accidents or road blockages can lead to severe disruptions. Drivers might unnecessarily alter their routes, leading to traffic congestion, increased travel time, and even actual accidents caused by sudden changes in traffic flow. The ability to detect and mitigate such attacks is vital for maintaining the trustworthiness and functionality of vehicular communication systems, ensuring that the safety of critical information remains reliable and effective in enhancing road safety. Figure 1 illustrates the scenario of a false alert attack in a vehicular network. In this scenario, a malicious vehicle sends a false alert message to other vehicles, indicating a non-existent accident on the road. Upon receiving the false alert, nearby vehicles may react by changing their routes or slowing down, causing traffic congestion and delays. This deceptive behavior can disrupt the normal flow of traffic, leading to potential accidents and safety hazards. Detecting and preventing false alert attacks is essential for maintaining the integrity and reliability of vehicular communication systems and safeguarding against malicious actors.

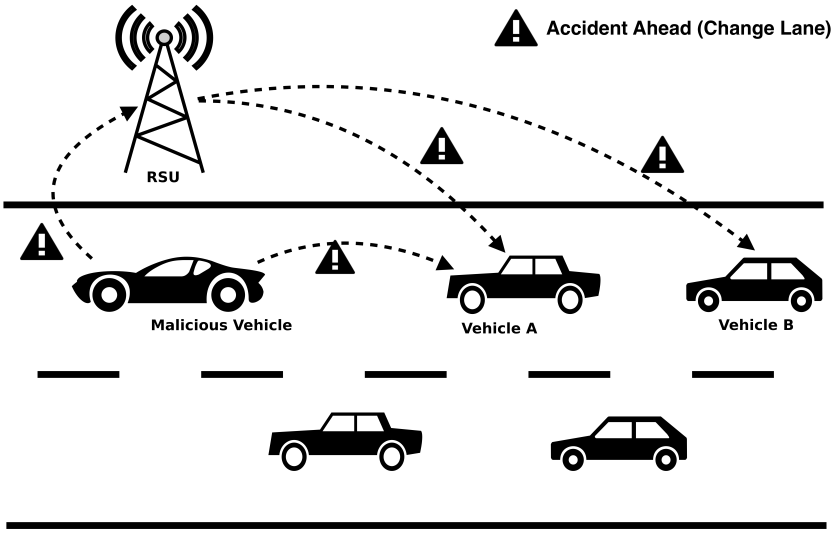


Figure 1. False alert attack in VANETs.

Detection and mitigation of this type of attack is particularly challenging due to the critical nature of the falsified information, which can lead to dangerous situations on the road. This study addresses the challenge by proposing to leverage ML approaches tailored to the VeReMiAP dataset. Our approach focuses on applying classical ML algorithms for dataset interpretation and feature engineering. The framework follows traditional ML pipelines and consists of the following stages:

- Data preparation;
- Data pre-processing;
- Feature engineering;
- Model training.

Figure 2 shows a flow diagram of the proposed framework, where each of the aforementioned four stages are outlined using using dashed lines. In the following sections, we discuss the functions of each module in more detail.

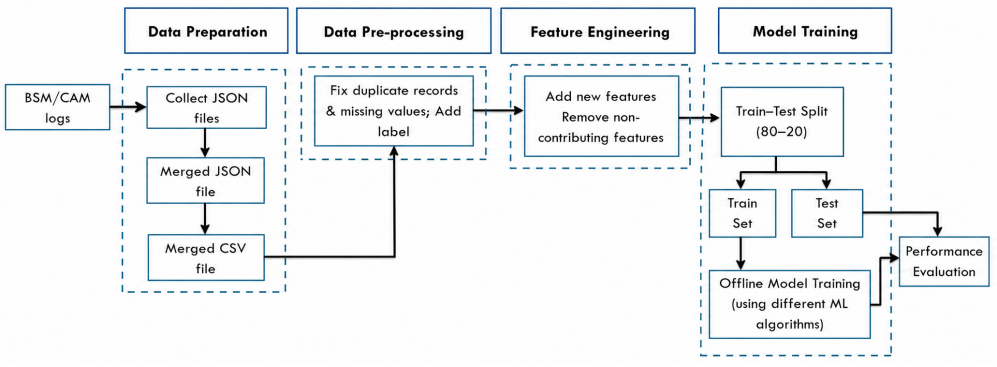


Figure 2. Flow diagram of false alert detection framework.

3.1. Data Preparation and Pre-Processing

VeReMiAP [6] is a simulated dataset designed to analyze false alert attacks. Each simulation run generates log files for each vehicle in the network. These log files are in JSON format, containing information on all the messages (BSMs and CAMs) transmitted by the vehicle, and are given as input to the data preparation module. Each vehicle in the simulation is classified as either "attacker" or "non-attacker". The individual log files are first merged into a single combined JSON file. This aggregated JSON file is then converted into CSV format, to facilitate integration with various data analysis and ML tools. Finally,

the consolidated CSV dataset is used for conducting exploratory data analysis (EDA) to identify any patterns and irregularities that may affect accurate detection of false alerts.

Prior to feature selection and model training, the dataset is first checked for constant columns, missing values, and duplicate records and any irrelevant columns are removed. Next, categorical variables like AttackLabel are encoded using one-hot encoding to convert them into numerical format. Finally, features containing composite values are segregated into multiple columns. For example, the column 'pos' was split into 'pos_x', 'pos_y', and 'pos_z', representing the x, y, and z components of the vehicle's position.

The dataset comprised 832,455 messages from 826 unique vehicles over a 3600 s timeframe, with 205 (24.82%) vehicles identified as attackers. A combined total of 366,668 BSMs and 465,787 CAMs were transmitted by all the vehicles. The accident reporting ratios for both attackers and non-attackers were similar, suggesting that attackers attempted to blend in with normal behavior. There is a significant imbalance (about 25% vs. 75%) between attacker vs. non-attackers vehicles in the simulated dataset, which should be taken into consideration to ensure that the detection and mitigation measures are robust.

Although the initial dataset contains both BSMs and CAMs, the BSMs do not contain any fields for alert notifications. Therefore, BSMs can never transmit alerts, and such packets are irrelevant for false alert detection. On the other hand, CAMs may or may not report alerts. CAMs that do not contain any alerts, clearly cannot report "false" alerts and are also irrelevant for false alert detection. Therefore, we removed all BSMs and CAMs without alert notifications from the dataset. As BSMs have much fewer columns than CAMs, including BSMs would also lead to a large number of null values resulting in a sparse dataset and making it more difficult to train the ML models. Thus, removing all messages without explicit alert notifications resulted in a smaller dataset and consequent reduction in training time and computational overhead without compromising the model's performance. This data pruning might lead to a higher false positive if the model is presented with all incoming BSMs/CAMs for prediction in real-world deployments. Therefore, it is recommended to add a filtering stage within the pre-processing block that filters out any irrelevant messages. The output of the filter will consist of CAMs with non-zero values in the eventID field. Hence, only the relevant messages will be forwarded to the false alert detection model for evaluation during actual deployment. Table 2 shows a comparison of the number of BSMs and CAMs before and after removing messages without alerts. We note that the percentage of attacker vehicles and the number of fake reports—as a percentage of all messages (both legitimate and fake) from all the vehicles—in the reduced dataset remain virtually unchanged compared to the original dataset.

3.2. Feature Engineering

Each CAM message can be decomposed into 27 original features, including the AttackLabel, as reported in Table 1. The data pre-processing and segregation step, where each composite feature, e.g., position, speed, acceleration, were split into multiple columns, resulted in a feature set of 63 features. We call this resulting feature set (before any feature engineering is carried out) FS_1. We performed initial model training and classification with FS_1, using a range of different algorithms. However, they did not yield high-quality results. Although, a few models achieved slightly over 80% accuracy, all the models performed poorly in terms of recall values, with scores ranging from 34 to 64%.

Table 2. Comparison of dataset metrics.

	Initial Values	**After Removing Non-Alert Messages**
Total Number of BSMs	366,668	0
Total Number of CAMs	465,787	45,484
Total Number of Messages (i.e., BSMs + CAMs)	832,455	45,484
Total Number of Unique Vehicles	826	616
Total Attacker Vehicles	205	153
Percentage of Vehicles that are attackers	24.82%	24.84%
Total Number of CAMs from Non-attacker	344,469	34,081
Total Number of CAMs from Attacker	121,318	11,403
Percent of Fake Reports	25.07%	25.07%

The initial feature set FS_1 contained a number of non-contributing features, such as message type, and z-coordinates of position, speed, and acceleration, which are always set to 0. As a first step towards improving performance, we refined the feature set by removing features that had very little or no contribution towards the final classification. Next, we created new features that aimed to capture underlying patterns and relationships within the data to help distinguish between attackers and non-attackers. Table 3 lists the 15 new features, along with their descriptions. Some of these new features focus on the vehicle's general characteristics, such as speed (spd) and the magnitude and direction of acceleration (acl_{mag}, acl_{dir}), which should have plausible values and be consistent with each other. In addition to the general features, we also added features that encapsulate the vehicle's behavior before and after reporting an accident. For example, after observing a road accident or other hazard, the expected behavior of a vehicle, in most cases, would be to slow down and/or change lanes. The new features, such as (spd_{10}, spd_{20}, spd_{100}) and (LI_{10}, LI_{20}, LI_{100}), attempt to capture this type of behavior.

Table 3. List of new features.

Feature	**Description**
spd	magnitude of vehicle speed $\sqrt{spd_x^2 + spd_y^2}$
acl_{mag}	magnitude of vehicle acceleration $\sqrt{acl_x^2 + acl_y^2}$
acl_{dir}	direction of acceleration $\theta_i^{ACL} = \text{atan2}\Big((\text{acl}_y)_i,\ (\text{acl}_x)_i\Big)$
hed_{dir}	direction of the vehicle's heading $\theta_i^{HED} = \text{atan2}\Big((\text{hed}_y)_i,\ (\text{hed}_x)_i\Big)$
$diff_{dir}$	difference between direction of the vehicle's heading and direction of acceleration $\Delta\theta_i = \theta_i^{ACL} - \theta_i^{HED}$
$acl_{overall}$	overall vehicle acceleration $\begin{cases} -\lvert ACL\rvert_i & \text{if } \lvert\Delta\theta_i\rvert \geq \frac{\pi}{2} \\ +\lvert ACL\rvert_i & \text{otherwise} \end{cases}$
spd_{10} (spd_{20}, spd_{100})	binary value indicating if the vehicle's speed decreased within 10 (20, 100) s of reporting an accident.
LI_{10} (LI_{20}, LI_{100})	binary value indicating if vehicle changed its lane, i.e., Lane Index value, within 10 (20, 100) s of reporting an accident.
$flag_h$	hazard flag indicating if the vehicle's hazardOccurrencePercentage is less than the average.
$flag_a$	acceleration flag indicating if the acl x avg and acl y avg is less than average, which in itself is the average of acl x and acl y for the sender vehicle.
$flag_t$	binary value indicating if the sender vehicle has been sending these accident messages more than the average vehicle.

At this stage of our work, we created new features based on an intuitive understanding of parameters that could be relevant to detecting false alerts, rather than on statistical analysis. An important next step for further developing our proposed model would be to conduct ablation experiments to measure the contribution of each of the features. For example, it is quite possible that we only need to keep only one feature from each of the groups (spd_{10}, spd_{20}, spd_{100}) and (LI_{10}, LI_{20}, LI_{100}) instead of including all 6 features.

Once the new features were engineered, the highly correlated features were removed from the dataset. The updated feature set, including the 15 engineered features, and after removing redundant and non-contributing features, is referred to as FS_2. Figure 3 shows the correlation matrix between the different features in FS_2. We see that the number of features having strong correlation with each other is relatively low, which helps to reduce problems with multicollinearity, increased complexity and unstable coefficients.

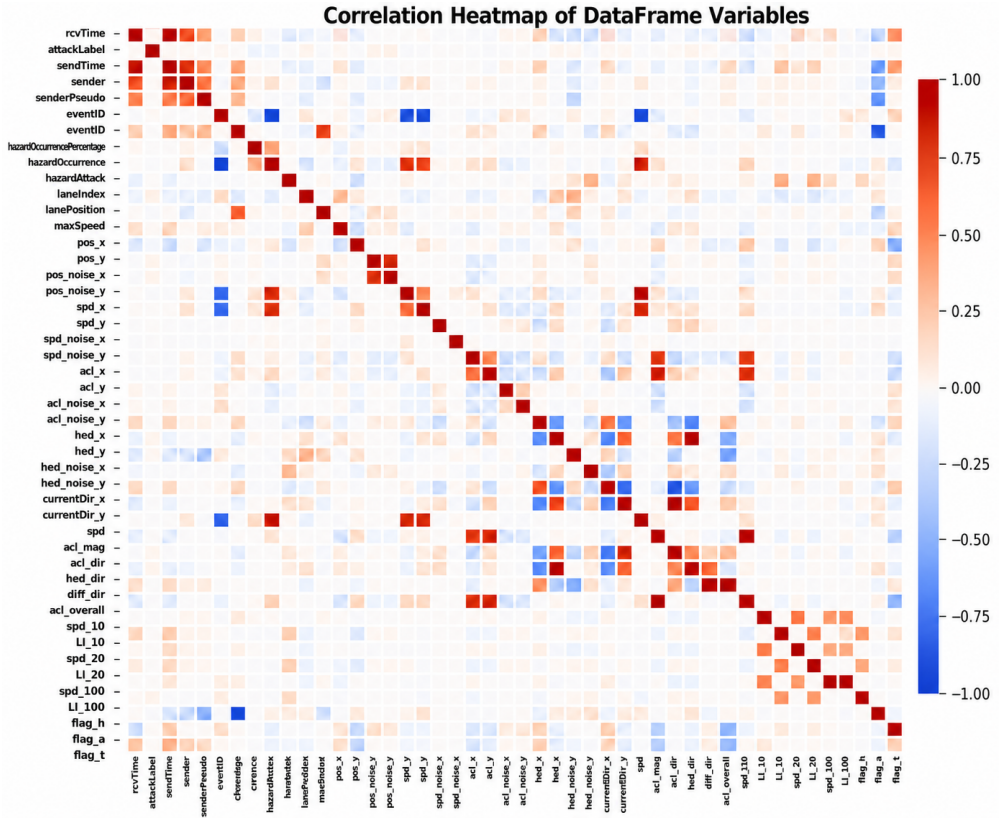


Figure 3. Heatmap of features.

Figure 4 shows the top 10 correlated features in FS_2 with the target variable. We note that several of the new engineered features are among the top 10 correlated features. Furthermore, even the highest correlation with the target variable is relatively low, with a value of 0.07, indicating that no single feature by itself would be a good predictor of a false alert.

3.3. Model Training

The machine learning approach involved splitting the dataset into 80% training and 20% testing sets. A variety of models were employed, including Decision Trees, Random Forest, Neural Networks, XGBoost, LSTM, LightGBM, CatBoost, and a Stacking model. Evaluation metrics such as accuracy, precision, recall, and F1 score were used to assess model performance.

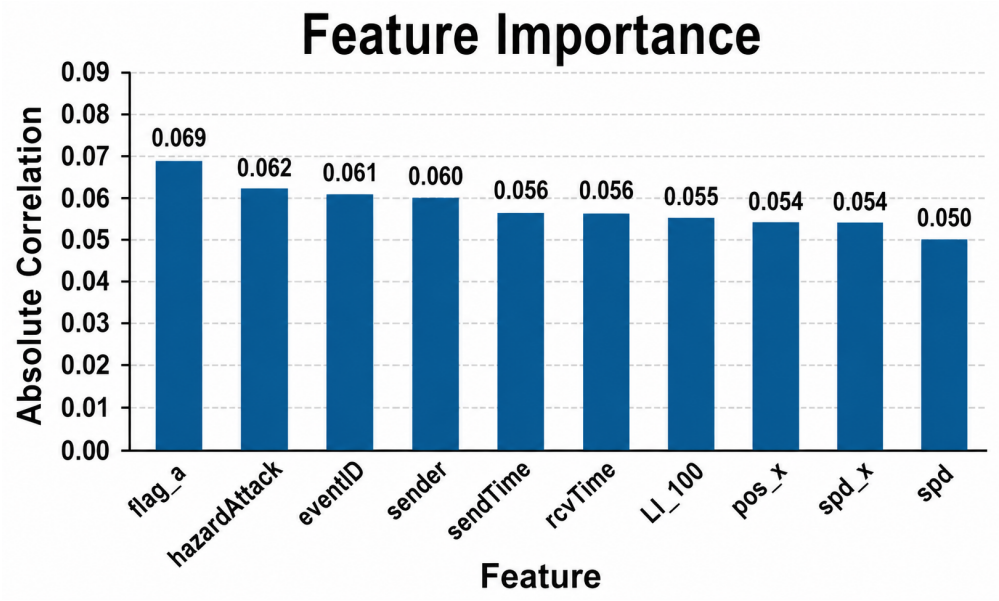


Figure 4. Top 10 correlated features with the target variable.

The choice of the meta-learner in the stacking models was crucial for optimizing performance. Initially, Decision Tree was used as the meta-learner due to its simplicity and interpretability. While the corresponding results were promising, we aimed for further enhancements. Subsequently, experiments were carried out with more advanced models, specifically LightGBM and XGBoost. XGBoost also demonstrated strong performance, due to its robust boosting framework that reduces overfitting and enhances generalization. LightGBM performed exceptionally well, primarily due to its ability to efficiently handle large datasets, its gradient-based optimization techniques, and its capacity to model complex interactions among base learners. The results for the different models are discussed in detail in Section 4. Finally, as the primary focus of the paper was to see if ML approaches can be effective for false alert detection, we simply used default parameters for the different algorithms. It is expected that proper selection of the hyperparameters can lead to further improvements in the performance of the models. Overall, the success of the stacking model with multiple learners highlights the potential of ensemble methods in addressing this challenging security issue in vehicular networks.

4. Performance Evaluation

In this paper, we have used the VeReMiAP dataset to train and evaluate the performance of our proposed ML models for detecting false alert attacks in VANETs. We note that we only consider messages that have been successfully received by the vehicle or RSU. Channel congestion or other adverse network conditions may affect successful delivery of some BSM/CAM packets. However, this would not affect the performance of the models, as they deal only with successfully received packets. The VeReMiAP dataset contains a 25:75 ratio of attack to non-attack messages. This disparity makes accuracy an inadequate metric for evaluating the performance of models, as they can easily predict the majority class and still achieve a high accuracy. Existing techniques for mitigating class imbalance, such as SMOTE, undersampling or class weights, typically distort the true data distribution, which can bias the probability calibration and cause the model to learn skewed decision boundaries. Introduction of synthetic instances that do not accurately represent the minority class can potentially lead to overfitting as well [39], while undersampling can result in significant data loss. On the other hand, it has been noted that ensemble learning-based methods such as XGBoost and LightGBM can handle class imbalances quite well without

the need for synthetic resampling [40]. Therefore, in this work we have decided to use the original data without any resampling and instead incorporate models that are less susceptible to class imbalance. We have also used different performance metrics, including precision, recall, and F1-score, rather than relying on just accuracy scores, in order to get a more balanced view of the performance.

4.1. Classification Results

With the goal of detecting false accident reports, we frame the problem as a binary classification task. Table 4 shows a confusion matrix for this task with a breakdown of the correct and incorrect predictions, where "Positive" indicates a false alert and "Negative" indicates a genuine alert message. Based on this, we define the four metrics for our evaluation as follows:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- F1 Score = $2 \times \left(\frac{Recall \times Precision}{Recall+Precision}\right)$

Table 4. Evaluation confusion matrix.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The dataset contains CAM messages with accident reports, some of which are false alerts. The classification task evaluated several algorithms, including Decision Trees, Random Forests, and Neural Networks. Table 5 shows the results for the models trained with feature set FS_1, before any feature engineering. Initial experiments with classical models yielded suboptimal results, with accuracy scores below 58% for Decision Trees, and with Random Forests and Neural Networks achieving an accuracy of 76%. The remaining models, including the stacking model (using XGBoost, CatBoost, Decision Tree, Random Forest, Logistic Regression), had slight improvements over Random Forest, with accuracies ranging from 81 to 83%. All the models struggled significantly with recall, with most achieving scores well below 50%. Only the stacking model performed slightly better, with a score of 64%. The poor recall and precision (for most models) scores led to unacceptable overall performance in terms of F1 Score for all models, with the Stacking Model achieving the highest F1 Score of 65% and all other models scoring below 60%.

Table 5. Performance of different models with FS_1.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Decision Tree	0.58	0.29	0.42	0.34
Random Forest	0.76	0.53	0.43	0.47
Neural Networks	0.76	0.55	0.45	0.50
XGBoost	0.81	0.71	0.43	0.53
LSTM	0.81	0.76	0.34	0.46
CatBoost	0.82	0.77	0.43	0.55
Stacking Model	0.83	0.68	0.64	0.65

Due to the relatively poor performance with FS_1, feature engineering techniques were applied to create more informative variables. Even though Decision Tree and Random Forest models did not perform very well with FS_1, we kept them for continued evaluation due to their relative simplicity. However, we removed both LSTM and Neural Networks from further consideration using FS_2. Since the Stacking Model had the best performance with FS_1, we decided to try several different stacking algorithms with FS_2, as outlined in Table 6.

Table 6. Identifiers and corresponding algorithms used with FS_2.

ID	Algorithm
M1	Decision Tree
M2	Random Forest
M3	XGBoost
M4	CatBoost
M5	LightGBM
	Stacking Models
M6	Stacking Model-Decision Tree (meta learner), CatBoost, XGBoost
M7	Stacking Model-Decision Tree (meta learner), CatBoost, XGBoost, Random Forest, Logistic Regression
M8	Stacking Model-Decision Tree (meta learner), CatBoost, LightGBM, Random Forest, Logistic Regression
M9	Stacking Model-Decision Tree, CatBoost, LightGBM (meta learner), Random Forest, Logistic Regression
M10	Stacking Model-Decision Tree, XGBoost, CatBoost, LightGBM (meta learner), Random Forest, Logistic Regression

Table 7 shows the performance of the different ML modes, using FS_2. For the stacking models, the meta-learners were chosen on a trial-and-error basis. We experimented with a few different meta-learners and selected the ones with the best performance, from among those tested. Apart from the accuracy, precision, recall, and F1 score, it is also important to consider the training and prediction times. Because, in real-world scenarios, the models must be trained and deployed quickly to ensure timely and effective detection of false alert attacks in VANETs. The training time represents the duration required to train the model on the dataset, while the prediction time indicates the time taken to make predictions on new data. These metrics are crucial for evaluating the models' computational efficiency and suitability for deployment in vehicular networks.

The last two stacking models (M9 and M10) achieved accuracies over 90% and also had the best overall performance, with F1 Scores of 84% and 83%, respectively. However, these two models also had the longest training times (over 76 s) and prediction times (around 8–9 ms). In general, although the stacking models had superior performance, they required longer training and took longer to make a prediction compared to most other models. For the non-stacking models, M4 (CatBoost) had an acceptable performance, achieving an accuracy of 88% and F1 Score of 72%. M4 also had significantly lower training time (5 s) and prediction time (0.5 ms) compared to the stacking models.

Table 7. Performance metrics and computational time of models using FS_2.

ID	Accuracy	Precision	Recall	F1 Score	Training Time (s)	Prediction Time (ms)
M1	0.86	0.75	0.65	0.70	0.7603	0.19
M2	0.84	0.71	0.60	0.65	7.9994	2.00
M3	0.87	0.85	0.60	0.71	0.4208	0.34
M4	0.88	0.88	0.61	0.72	5.0368	0.52
M5	0.86	0.90	0.50	0.64	0.9396	0.86
M6	0.86	0.74	0.71	0.73	34.2695	0.96
M7	0.88	0.78	0.74	0.76	73.6790	3.60
M8	0.88	0.78	0.75	0.76	75.7822	7.50
M9	0.92	0.86	0.80	0.84	77.7638	9.00
M10	0.92	0.87	0.79	0.83	76.5137	7.90

Figure 5 compares the overall performance (in terms of F1 Scores) of the different models when using FS_1 versus FS_2. We note that models trained using FS_2 consistently perform better, regardless of the training algorithm. The improvements range from about 12% for the stacking model to over 100% for the Decision Tree model, with average improvement of about 44%.

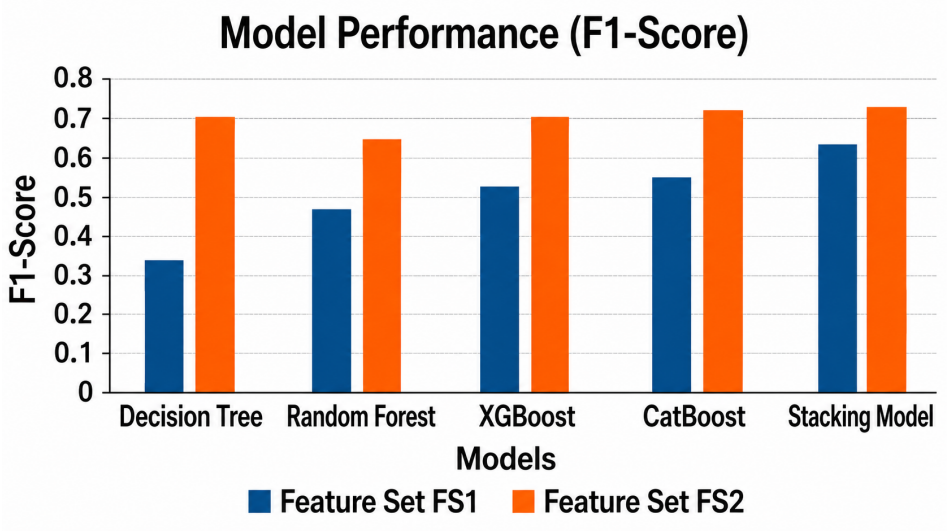


Figure 5. Comparison of F1-Scores using FS_1 vs. FS_2.

To evaluate the impact of the feature sets, we conducted a series of paired-samples *t*-tests comparing the classification metrics for FS1 and FS2 across four evaluation criteria: Accuracy, Precision, Recall, and F1 Score. FS2 achieved statistically significant improvements ($p < 0.05$) in terms of precision, recall and F1 score, with the most reliable improvement being for recall ($p = 0.001$). While FS2 does achieve a higher average score for overall Accuracy, this specific improvement falls just short ($p = 0.073$) of being statistically significant.

4.2. Model Selection for Different Scenarios

The selection of a machine learning model for detecting false alert attacks in VANETs is contingent upon the specific scenario and the available computational resources. This consideration ensures that the chosen models can operate efficiently and effectively within

the given constraints. Table 8 provides our recommendations of suitable models for different scenarios, categorized by available computational power and the deployment component. The models are evaluated based on their performance metrics, training and prediction times, and computational requirements.

Table 8. Model Deployment and Usage

Computation Power	Deployment Component	Usage	Models
High	RSUs	Training, Prediction	M9, M10
Moderate	Vehicles	Prediction	M7
Low computation power	Vehicles	Prediction	M6
Very Low computation power	Vehicles	Prediction	M1, M4

For components with high computational power, such as RSUs, models M9 and M10 are recommended. These components can handle both training and prediction tasks due to their robust computational capabilities. In vehicles with moderate computational power, model M7 is ideal for prediction tasks, providing accurate predictions while efficiently managing computational loads. Vehicles with low computational power are better suited for simpler models like M6, designed specifically for prediction tasks and optimized for limited processing capabilities. For vehicles with very low computational power, models M1 and M4 are recommended.

Considering the training aspects, RSUs should be preferred for training these models. Whereas, for prediction, the trained models can be deployed on the vehicles. This way, the models can be trained on a large dataset and then be used for real-time predictions.

4.3. Comparison with Existing Approaches

The false alert attacks have been introduced quite recently, and as such, there is very little previous work available for comparison. The work in [35] was the only relevant work for comparison, using a standard publicly available dataset that we could find in the literature. The approach focuses on examining vehicle behaviors during false alert attacks to identify hazardous phases. Table 9 shows that the proposed model achieves an 8% improvement in detection accuracy compared to [35], using the same VeReMiAP dataset. The work in [35] does not report the precision and recall scores, and are marked as 'X' in Table 9. Although accuracy and other metrics for false alert detection are reported in [33,34], they were evaluated using a different dataset. Therefore, a direct comparison would not be meaningful.

Table 9. Performance Comparison with Neural Network Approach

	Dataset	Approach	Accuracy	Precision	Recall	F1 Score
Paper [35]	VeReMiAP	Neural Networks	84%	X	X	X
Proposed Approach	VeReMiAP	Classical ML	92.1%	87.57%	80.35%	83.8%

We note that the applicability of the results discussed in this section may be limited due to its reliance on simulated vehicular environments rather than field-tested, real-world Vehicular Ad-Hoc Network (VANET) deployments. Additionally, the results may not be generalizable when applied across traffic densities, attacker ratios, or simulation scenarios that vary significantly from the simulated environment. Use of only a single dataset may result in features that are specifically tailored to the current data. Therefore, while

the evaluation clearly demonstrates the advantages of ML-based misbehavior detection approaches for false alerts, the actual improvement over non-ML approaches may vary depending on the pattern of attacks and the specific traffic scenarios. Finally, the current problem formulation focuses on classification using a pre-filtered dataset rather than an end-to-end implementation of a false alert detection system in realistic VANET environments.

5. Conclusions

In this paper we present a novel classical machine learning framework for detecting false alert messages in Vehicular Ad Hoc Networks (VANETs). A carefully engineered feature set that captures vehicle kinematics and suspicious behavioral patterns beyond the original dataset attributes was used to significantly improve model performance. We designed and evaluated multiple ML models, and although most models achieved acceptable accuracy, consistent challenges in recall scores indicated the difficulty of identifying true false alerts. The gradient boosting ensemble algorithms (XGBoost, CatBoost, and LightGBM) improved accuracy and precision, while the proposed stacking approaches achieved superior overall performance. Compared to existing techniques using the same VeReMiAP dataset, our framework achieved significant improvements. We also analyzed the computational trade-offs of different models across deployment scenarios. This work provides a baseline for ML-based false alert detection in VANETs, combining robust detection with deployment-aware model selection.

Future work to extend this study will focus on rigorous feature analysis, including ablation studies, to evaluate the contribution and importance of individual features, as well as hyperparameter tuning strategies to optimize model performance across different algorithms. Calibration using diverse datasets will be needed to ensure that the features are not simply tailored to the current dataset or contain artifacts introduced during the dataset generation process. The proposed models can be further augmented by integrating traffic anomaly detection techniques and vehicle consensus mechanisms as part of the detection framework. The reliance on a single simulated dataset can be mitigated through the use of a scalable parallel simulator framework, such as [41], that can generate more realistic and computationally efficient scenarios for evaluating false alert attacks in VANETs. Additionally, transitioning from simulated datasets to real-world vehicular data will be essential to validate the practical applicability and generalizability of the proposed framework in real deployment settings.

Author Contributions: Conceptualization, A.K., I.S. and A.J.; methodology, A.K.; software, A.K.; validation, A.K., I.S. and A.J.; investigation, A.K., I.S. and A.J.; writing—original draft preparation, A.K.; writing—review and editing, A.K., I.S. and A.J.; supervision, I.S. and A.J.; funding acquisition, I.S. and A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSERC DG, Grant# RGPIN-2022-03254.

Data Availability Statement: Data is contained within the article.

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

References

1. An, S.; Lee, B.H.; Shin, D.R. A survey of intelligent transportation systems. In Proceedings of the 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, Bali, Indonesia, 26–28 July 2011; pp. 332–337.
2. Rashid, S.A.; Audah, L.; Hamdi, M.M. Intelligent Transportation Systems (ITSs) in VANET and MANET. In *Biologically Inspired Techniques in Many Criteria Decision Making: Proceedings of BITMDM 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 667–675.
3. *SAE J2735_202409* ; V2X Communications Message Set Dictionary. Society of Automotive Engineers International: Warrendale, PA, USA, 2024.

4. *ETSI EN 302 637-2 V1.4.1*; Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. European Telecommunications Standards Institute (ETSI): Sophia Antipolis, France, 2019.
5. Sumra, I.A.; Sellappan, P.; Abdullah, A.; Ali, A. Security issues and challenges in MANET-VANET-FANET: A survey. *EAI Endorsed Trans. Energy Web* **2018**, *5*, e16. [CrossRef]
6. Abdelmaguid, M.A.; Hassanein, H.S.; Zulkernine, M. A VeReMi-based Dataset for Predicting the Effect of Attacks in VANETs. In Proceedings of the Int'l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, Montreal, QC, Canada, 30 October–3 November 2023; pp. 175–179.
7. Parkinson, S.; Ward, P.; Wilson, K.; Miller, J. Cyber threats facing autonomous and connected vehicles: Future challenges. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2898–2915. [CrossRef]
8. Bouchouia, M.L.; Labiod, H.; Jelassi, O.; Monteuuis, J.P.; Jaballah, W.B.; Petit, J.; Zhang, Z. A survey on misbehavior detection for connected and autonomous vehicles. *Veh. Commun.* **2023**, *41*, 100586. [CrossRef]
9. Nandy, T.; Noor, R.M.; Kolandaisamy, R.; Idris, M.Y.I.; Bhattacharyya, S. A review of security attacks and intrusion detection in the vehicular networks. *J. King Saud-Univ.-Comput. Inf. Sci.* **2024**, *36*, 101945. [CrossRef]
10. Nagarajan, J.; Mansourian, P.; Shahid, M.A.; Jaekel, A.S.; Ikjot.; Zhang, N.; Kneppers, M. Machine Learning based intrusion detection systems for connected autonomous vehicles: A survey. *Peer-to-Peer Netw. Appl.* **2023**, *16*, 2153–2185. [CrossRef]
11. Habehh, H.; Gohel, S. Machine Learning in Healthcare. *Curr. Genom.* **2021**, *22*, 291–300. [CrossRef] [PubMed]
12. Wu, X.; Wang, H.; Zhang, Y.; Zou, B.; Hong, H. A Tutorial-Generating Method for Autonomous Online Learning. *IEEE Trans. Learn. Technol.* **2024**, *17*, 1532–1541. [CrossRef]
13. Zeleznikow, J. The benefits and dangers of using machine learning to support making legal predictions. *WIREs Data Min. Knowl. Discov.* **2023**, *13*, e1505. [CrossRef]
14. Sun, Y.; Peng, M.; Zhou, Y.; Huang, Y.; Mao, S. Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3072–3108. [CrossRef]
15. Zhu, R.; Li, W.; Boukerche, A.; Yang, Q. Energy-Aware DRL-Based Dual-Perception Fountain Codes for Resource-Constrained UASNs. *IEEE Trans. Sustain. Comput.* **2026**, *11*, 111–122. [CrossRef]
16. Codecá, L.; Frank, R.; Faye, S.; Engel, T. Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 52–63. [CrossRef]
17. van der Heijden, R.W.; Lukaseder, T.; Kargl, F. VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs. *arXiv* **2018**, arXiv:1804.06701. [CrossRef]
18. Kamel, J.; Ansari, M.R.; Petit, J.; Kaiser, A.; Ben Jemaa, I.; Urien, P. Simulation Framework for Misbehavior Detection in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 6631–6643. [CrossRef]
19. Kamel, J.; Wolf, M.; Van Der Hei, R.W.; Kaiser, A.; Urien, P.; Kargl, F. Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
20. Grover, J.; Laxmi, V.; Gaur, M.S. Misbehavior detection based on ensemble learning in vanet. In Proceedings of the International Conference on Advanced Computing, Networking and Security, Surathkal, India, 16–18 December 2011; pp. 602–611.
21. So, S.; Sharma, P.; Petit, J. Integrating plausibility checks and machine learning for misbehavior detection in VANET. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 564–571.
22. Singh, P.K.; Gupta, S.; Vashistha, R.; Nandi, S.K.; Nandi, S. Machine learning based approach to detect position falsification attack in vanets. In Proceedings of the International Conference on Security & Privacy, San Francisco, CA, USA, 19–23 May 2019; pp. 166–178.
23. Gyawali, S.; Qian, Y. Misbehavior detection using machine learning in vehicular communication networks. In Proceedings of the ICC 2019–2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
24. Sharma, P.; Liu, H. A Machine-Learning-Based Data-Centric Misbehavior Detection Model for Internet of Vehicles. *IEEE Internet Things J.* **2020**, *8*, 4991–4999. [CrossRef]
25. Shams, E.A.; Rizaner, A.; Ulusoy, A.H. Trust aware support vector machine intrusion detection and prevention system in vehicular ad hoc networks. *Comput. Secur.* **2018**, *78*, 245–254. [CrossRef]
26. Karagiannis, D.; Argyriou, A. Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning. *Veh. Commun.* **2018**, *13*, 56–63. [CrossRef]
27. Oucheikh, R.; Fri, M.; Fedouaki, F.; Hain, M. Deep Anomaly Detector Based on Spatio-Temporal Clustering for Connected Autonomous Vehicles. In *Proceedings of the International Conference on Ad Hoc Networks*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 201–212.
28. U.S. Department of Transportation. Wyoming CV Pilot Basic Safety Message One Day Sample. Available online: https://catalog.data.gov/dataset/wyoming-connected-vehicle-pilot-basic-safety-message-visualization (accessed on 13 June 2026).

29. Aneja, M.J.S.; Bhatia, T.; Sharma, G.; Shrivastava, G. Artificial intelligence based intrusion detection system to detect flooding attack in VANETs. In *Handbook of Research on Network Forensics and Analysis Techniques*; IGI Global: Hershey, PA, USA, 2018; pp. 87–100.
30. Kamel, J.; Haidar, F.; Jemaa, I.B.; Kaiser, A.; Lonc, B.; Urien, P. A misbehavior authority system for sybil attack detection in c-its. In *Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*; IEEE: New York, NY, USA, 2019; pp. 1117–1123.
31. Zeng, Y.; Qiu, M.; Zhu, D.; Xue, Z.; Xiong, J.; Liu, M. Deepvcm: A deep learning based intrusion detection method in vanet. In *Proceedings of the 2019 IEEE 5th International Conference on Big Data Security on Cloud (BigDataSecurity)*; IEEE: New York, NY, USA, 2019; pp. 288–293.
32. Liu, J.; Yang, W.; Zhang, J.; Yang, C. Detecting false messages in vehicular ad hoc networks based on a traffic flow model. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720906390. [CrossRef]
33. Ghaleb, F.A.; Zainal, A.; Maroof, M.A.; Rassam, M.A.; Saeed, F. Detecting bogus information attack in vehicular ad hoc network: A context-aware approach. *Procedia Comput. Sci.* **2019**, *163*, 180–189. [CrossRef]
34. Zhang, Y.; Cheong, C.; Li, S.; Cao, Y.; Zhang, X.; Liu, D. False message detection in Internet of Vehicle through machine learning and vehicle consensus. *Inf. Process. Manag.* **2024**, *61*, 103827. [CrossRef]
35. Abdelmaguid, M.A.; Hassanein, H.S.; Zulkernine, M. Attack endgame: Proactive security approach for predicting attack consequences in VANET. In Proceedings of the ICC 2023—IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023; pp. 3762–3767.
36. Abbasi, H.I.; Voicu, R.C.; Copeland, J.A.; Chang, Y. Cooperative bsm architecture to improve transportation safety in vanets. In Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 1016–1022.
37. Breu, J.; Brakemeier, A.; Menth, M. A quantitative study of Cooperative Awareness Messages in production VANETs. *EURASIP J. Wirel. Commun. Netw.* **2014**, *2014*, 98. [CrossRef]
38. Karhana, A.; Saini, I.; Jaekel, A. Machine Learning Approach to False Alert Attack Detection in Vehicular Adhoc Networks. In Proceedings of the 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2025; pp. 1–7.
39. Alkhawaldeh, I.M.; Albalkhi, I.; Naswhan, A.J. Challenges and limitations of synthetic minority oversampling techniques in machine learning. *World J. Methodol.* **2023**, *13*, 373–378. [CrossRef] [PubMed]
40. Reddit MachineLearning Community. When to Use SMOTE When Dealing with Rare Events. 2022. Available online: https://www.reddit.com/r/MachineLearning/comments/vciipu/d_when_to_use_smote_when_dealing_with_rare_events/ (accessed on 10 June 2026).
41. Grinberg, I.; Wiseman, Y. Scalable Parallel Simulator for Vehicular Collision Detection. *Int. J. Veh. Syst. Model. Test.* **2013**, *8*, 119–144. [CrossRef]

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