Journal of Scientific & Industrial Research
Vol. 85, June 2026, pp. 520-530
DOI:10.56042/jsir.v85i6.16218

D-MNR: Dynamic Malevolent Node Recognition in Vehicular Communications — A Machine Learning Approach

Y Sarada Devi* & Roopa M

Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, SRM Institute of Science & Technology, Ramapuram Campus, Tamil Nadu, Chennai, 600 089, India

Received 05 January 2025; revised 15 May 2026; accepted 03 August 2026

For Vehicular Ad Hoc Networks (VANETs) to be secure and private, it is essential to identify hostile activity. The problem of detecting black hole attacks, which seriously jeopardize the Dynamic Source Routing (DSR) protocol, is discussed in this study. While effective for static data, traditional machine learning-based approaches find it difficult to adjust to the constant stream of real-time data in VANETs. We use incremental classifiers, which can analyze data dynamically for prompt attack detection, to get around this issue. In particular, we assess how well Adaptive Random Forest (ARF) and K-Nearest Neighbours (KNN) identify black hole attacks. In order to capture important characteristics that define black hole behaviour, the data used for attack detection is simulated under controlled conditions. Key performance parameters, such as accuracy, precision, recall, F1-score, and training/testing timeframes, are used to evaluate the classifiers. Despite requiring longer training and testing durations, the results reveal that ARF performs better than KNN across all criteria, indicating greater detection capabilities for black hole attacks. The results emphasize the usefulness of ARF for enhancing security in dynamic network environments and demonstrate the potential of incremental online learning techniques for continuous and real-time threat detection in VANETs. This study suggests a scalable approach for real-time cyber security in automotive networks and provides fresh insights into improving threat detection techniques.

Keywords: Adaptive random forest, Black hole attack, Intrusion detection, K-nearest neighbours, Simulation of urban mobility

Introduction

Vehicular Ad Hoc Networks (VANETs) are a subset of Mobile Ad Hoc Networks (MANETs) that are designed to allow communication between vehicles that are equipped with advanced communication devices and integrated features.[1] VANETs are of paramount importance for enhancing traffic efficiency, road safety and infotainment services. VANETs enable vehicles to communicate with each other and with infrastructure to reduce accidents, optimize traffic flow and provide drivers with real-time data. VANETs, however, are susceptible to a wide range of security threats as any other communication system, and attacks such as Denial of Service (DoS), black hole and wormhole attacks are particularly dangerous to the functionality and privacy of the network.

Machine Learning (ML) has become a prominent technique for detecting these attacks in VANETs, due to its ability to analyze massive datasets and identify patterns that indicate malicious actions. Several studies have proposed attack detection solutions for VANETs using machine learning. These studies emphasize the effectiveness of Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forest (RF) algorithms in detecting and mitigating threats.[2] However, these methods face a lot of challenges in the dynamic environment of VANETs, where the network topology is always changing and attack patterns are changing with time. Models trained on fixed datasets are often unable to adapt to new and emerging threats, leading to reduced accuracy and effectiveness in real-world applications.

This limitation indicates the need for more flexible solutions able to learn continuously and update their models in real time as new data become available. Incremental online learning algorithms can tackle this problem, enabling the model to learn from data in an incremental way and to adapt to new patterns without retraining on large datasets. The algorithms offer the scalability and flexibility needed for ongoing monitoring and attack detection in VANETs, allowing for prompt and accurate reactions to emerging threats.

The main objective of this work is to investigate the application of incremental online learning

*Author for Correspondence
E-mail: sy9149@srmist.edu.in

algorithms for attack detection in VANETs, especially black hole attacks, which are one of the most severe threats to the Dynamic Source Routing (DSR) protocol. A black hole attack is an attack where a malicious node falsely claims to be the optimal path and intercepts and drops data packets. This results in severe data loss and network inefficiency. This attack can disrupt communication and degrade network performance, and the widely adopted DSR protocol in VANETs is highly vulnerable to it.

To address this challenge, the study employs a methodology based on realistic VANET scenarios simulated with the Network Simulator 3 (NS-3) and Simulation of Urban Mobility (SUMO) simulators. The data collected from these simulations includes important features related to black hole attack detection such as the number of dropped packets, throughput and node behaviour patterns. The two classifiers, K-Nearest Neighbours (KNN) and Adaptive Random Forest (ARF), are evaluated over time on their performance using the key metrics, such as precision, accuracy, F1-score, and recall. The study also compares the computational complexities of both classifiers by examining their training and testing times to evaluate their suitability for real-time applications.

By concentrating on incremental learning techniques for attack detection in VANETs, especially in real-time scenarios, this work will close a gap in the existing literature. Machine learning methods have been applied to VANET security in earlier research. The majority of them have concentrated on offline learning or static datasets, which ignore the dynamic nature of attacks. In order to detect black hole attacks in dynamic VANET environments, the suggested study offers a scalable and effective method by examining continuous learning approaches that can manage real-time data streams.

A major subfield of Artificial Intelligence (AI), machine learning has drawn a lot of interest due to its potential for a variety of real-world uses, including speech recognition, image processing, autonomous cars, fraud detection, and attack detection.[3] In the context of VANETs, a variety of ML-based techniques have been put forth to address security issues, especially in attack detection.[4] These initiatives demonstrate how ML is becoming more and more crucial to preserving communication integrity in VANETs.

A number of studies suggested novel approaches to VANET attack detection. One intriguing study highlights the need to distinguish between congestion and deliberate jamming events and suggests a unique technique to detect blocking in wireless vehicle networks.[5] By examining safety messages and locating rogue nodes, another study[6] seeks to discover fake location information in VANETs. Additionally, research utilizing algorithms like Support Vector Machines (SVM) and K-Nearest Neighbours (KNN) for attack categorization has shown the value of applying ML approaches to misbehaviour detection in VANETs.[7,8]

However, despite the fact that these studies are quite effective at identifying specific kinds of attacks, they are not well suited to the rapidly evolving VANETs, which produce big-data environments for the detection of new attacks. Using Random Forest (RF) for high accuracy detection, some work has been done on multi-classification of attacks.[9–13] Although hybrid models of deep learning[14,15] and machine learning and optimization[16] have been proposed for enhanced intrusion detection systems, their static techniques make them unsuitable for real-time dynamic attacks. Additionally, while some research has addressed machine learning for specific attacks, such as position falsification and multiple attacks,[17–21] they fall short of the necessary adaptive learning systems.

Real-time incremental machine learning models for VANETs that adjust to data and threats as they enter systems and environments are lacking in the current research. In the presence of dynamic situations, many approaches struggle and fail with scalability and performance, particularly for black hole attacks. Black hole assaults are detrimental, particularly to the DSR routing protocol. This research concentrates on using incremental learning to detect black hole assaults in VANETs in real time. With an emphasis on K-Nearest Neighbours (KNN) and Adaptive Random Forest (ARF) techniques, the study provides a unique method for identifying data streams in order to enhance detection and reaction times in a dynamic and time-critical setting. To provide an effective way to safeguard VANETs, the methods discussed concentrate on incremental data streams and real-time black hole attack patterns.

VANET security research has recently been more focused on adaptive and online learning based intrusion detection techniques to handle the dynamic nature of vehicular environments. Federated Learning (FL) based intrusion detection frameworks offer improved privacy preservation and scalability by

enabling distributed model training without the need to share raw vehicular data. Likewise, adaptive fusion and ensemble learning models have demonstrated improved detection capability against evolving cyber threats through collaborative and real-time learning mechanisms. These approaches emphasize adaptability and robustness but generally impose additional communication and computational load on highly mobile VANET scenarios.

The proposed D-MNR (Dynamic Malevolent Node Recognition) framework employs incremental online learning algorithms, particularly Adaptive Random Forest (ARF), for efficient detection of black hole attacks in DSR-based vehicular communications as compared to recent approaches. Compared with traditional batch learning methods, the proposed model is more suitable for real-time intrusion detection as it adapts to incoming network data in an online manner. Experimental results show that ARF has better accuracy, precision, recall and F1-score than KNN and can adapt to a dynamic network environment. Furthermore, recent studies have investigated hybrid deep learning, federated learning, and ensemble-based intrusion detection frameworks for VANET security, indicating future opportunities to improve the presented model through distributed and hybrid learning strategies.[22,23]

Objectives of This Research

1. To study incremental online learning algorithms to detect attacks in VANETs at the time of occurrence.
2. To obtain the classification efficiency of the KNN and ARF classifiers against black hole attacks from a practical simulation-generated dataset.
3. To assess the performance of these classifiers based on precision, accuracy, F1-score and recall.
4. To compare the computational complexities of the KNN and ARF models in relation to their training and testing times, and their suitability for real-time applications.

Through the study of the restrictions of conventional machine learning approaches in vehicular ad hoc networks and the usage of incremental online learning, this study opens up new ways of securing vehicular networks against attacks. Such developments with practical application would ensure better resilience of networks and a higher attack detection rate.

Methodology

To initiate this section, the calculations of the produced method for attack detection are first introduced, which is further followed by incremental online machine learning. The data collection process is described in detail, including experimental designs, the environment used for the simulation and the features selected for measurement. The use of incremental online methods is discussed in detail, along with the tools and frameworks.

Incremental Online Learning

There has been an increased focus on incremental online learning recently, particularly in the situation of learning from streams of data, where information keeps coming in. Applications such as the Internet of Things, attack detection, spam filtering, and time series forecasting make this clear.

Conventional machine learning techniques usually require significant historical data collection, where the model is trained on a specified set of data and subsequently tested or validated on a test set. This is an effective method where all the data is available ahead of time to train the model on a large fixed dataset. Unlike these algorithms that utilize fixed data sets, the methods discussed here develop dynamic models that are trained continuously, rather than operating on a fixed dataset that is processed all at a time. Consider S the set of all data having examples represented by Eq. (1):

$$(\rightarrow n_i, m_i),\ k = 1, 2, 3, \ldots, x, \ldots \quad \ldots (1)$$

where, x represents a timestamp, $\rightarrow n_i$ is a feature vector of S and m_i is the corresponding target.

The aim is to continually acquire knowledge from incoming data and consistently update the model. The fundamental distinction between online learning and traditional learning lies in the way data is accessible. In traditional learning, the data remains stationary and is entirely available, whereas online learning processes data instances sequentially over time. Incremental online learning embodies various key characteristics. It enables the utilization of data as it becomes available, ensuring consistently up-to-date models rather than processing the entire dataset at once. This attribute proves particularly advantageous in active applications like VANETs, where statistics are created in a flooding manner. The necessity to develop minor additions of information for learning distinguishes incremental online learning from old-style ML techniques, which demand significant

computing, especially when handling extensive datasets. The capability to handle streaming data in real time renders incremental online learning highly advantageous in research areas where data generation is continuous and necessitates real-time analysis. The adaptability of incremental online learning allows it to adjust to changing data patterns over time.

Suggested Approach

The fundamental concept of the proposed method turns on applying real-time data from monitoring VANET node behaviour and employing incremental online learning algorithms to train an online model. The updated model generated through this incremental progression is applied for the organization of nodes as either normal or malicious. This module engages in the online classification of unlabelled data, utilizing the continually updated and trained model to classify incoming data in real time. The online sorting enables the detection of attacks by categorizing patterns indicative of malicious activity. If the pattern aligns as a known one, an alarm is generated. The projected way, as shown in Fig. 1, for detecting the attack contains a group phase of data exploiting a system designed to observe behavioural parameters and metrics within each node. This incorporates seizing the most pertinent features to discriminate between black hole nodes and normal nodes in the network.

Primary Steps of the Method Proposed

Preliminary Training

An original model is trained on primary data to establish a base before attracting learning. This model serves as the preliminary stage for the ongoing incremental learning process.

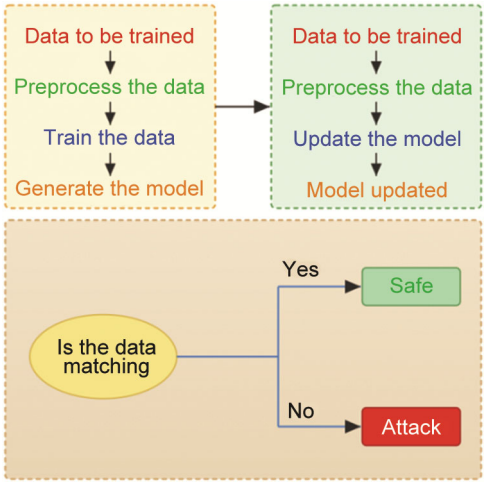


Fig. 1 — Proposed method

Training the Model

The model is updated incrementally by meeting out each instance that is newly accessible at present. This incremental learning procedure occurs continuously each time original data becomes accessible. Instead of training the model from scratch on the entire dataset, it is refined using new data. The reorganized model is then employed for node classification as either malicious or normal.

Detection of Attack

The real-time classification enables the timely detection of attacks by recognizing patterns indicative of malicious behaviour. If the identified pattern corresponds to a known attack, an alarm is activated.

Recording various metrics by collecting the data are the stages involved to notice the attacks. Subsequent subsections provide technical details on data collection and preprocessing, covering aspects such as the network simulation environment and the selection of pertinent features for attack detection by collecting data at the location.

In the initial scenario, 70 vehicles are engaged, forming 20 random source-to-destination pairs that generate a bitrate with packets of secure size. The routing of packets is facilitated by the DSR protocol, assuming all nodes to be legitimate vehicles. Notably, this simulation does not incorporate any black hole nodes.

In the subsequent simulated scenario, the DSR routing protocol is implemented, introducing black hole attacks. The DSR protocol, as shown in Fig. 2, is a widely used routing protocol in wireless ad hoc networks. It belongs to the category of on-demand, reactive routing protocols, which means that routes are only determined when needed for communication. DSR is specifically designed for multi-hop wireless ad hoc networks, where nodes communicate with each other without the need for a fixed infrastructure.

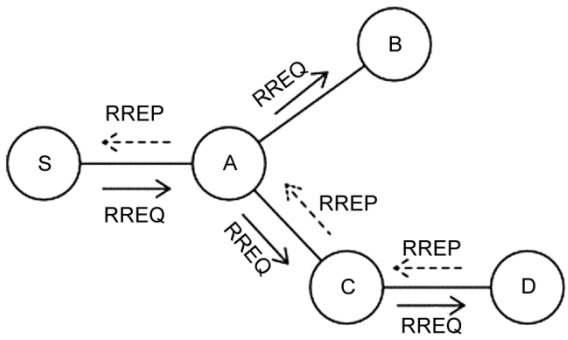


Fig. 2 — DSR protocol

Key Features of the DSR Protocol

On-Demand Routing

DSR operates on a reactive basis, establishing routes only when there is an actual need for communication. This helps in reducing the overhead associated with maintaining routing information for all possible destinations in the network.

Source Routing

One distinctive feature of DSR is its use of source routing. In source routing, the complete route from the source node to the destination node is included in the packet header. Each intermediate node along the route forwards the packet based on the information contained in the header.

Route Discovery

When a node needs to communicate with a destination for which it does not have a route, it initiates a route discovery process. During this process, the source node broadcasts a Route Request (RREQ) packet that traverses the network seeking the destination. Nodes receiving the RREQ may reply with a Route Reply (RREP) packet if they have a route to the destination, or forward the RREQ further.

Caching of Routes

DSR supports route caching, where intermediate nodes can store discovered routes for future use. This helps in reducing the latency and overhead associated with repeated route discoveries for the same destination.

Loop Detection

The protocol incorporates mechanisms to prevent and detect routing loops. Each node maintains a list of recently visited nodes to avoid revisiting the same nodes during route discovery.

Flexibility

DSR is known for its flexibility, making it suitable for various ad hoc network scenarios. It can adapt to changes in network topology and is resilient to node mobility.

Although DSR has the benefits of low overhead and high flexibility in adapting to a dynamic network, it also has some drawbacks, such as high overhead in large networks and susceptibility to route discovery storms. There are still many ways to enhance and modify the DSR protocol to overcome these issues and optimize it for various situations. In this case, 1–4 nodes are specifically chosen to be assailants, and are built at different times to perform black hole attacks. All other nodes are trusted and communicate with other vehicles in the network in a valid manner. A monitoring system is designed to collect the parameters of the vehicles' behaviour and various metrics in every vehicle node during the simulation, both in normal and in black hole attack situations. This is a full data collection to gather relevant data that is important for separating out black hole and normal nodes. Following this, a brief description is given about the characteristics of the DSR protocol and its vulnerability to black hole attacks. Thus, the malicious node fools the source into believing that route discovery has finished. While recording different behavioural parameters and metrics, the following parameters are important.

Behavioural Parameters Recorded

Analysis of Behavior in Routing

This comprises chasing complete routing control packets in DSR and nursing specific packets like RRP (Route Reply), RRQ (Route Request), and RER (Route Error). These parameters contribute to attack detection and are abridged in the features CRRP, CRRQ, CT packets and CRER.

Analysis of Traffic

This entails monitoring traffic features and keeping a record of the number of bytes communicated by each node.

Monitoring Dropping Proportion

Perceiving the packet dropping rate aids in identifying nodes that drop selectively or abandon arriving packets, indicative of malicious behaviour.

Monitoring Throughput

The black hole attack classically results in a significant drop in data, affecting throughput. Thus, monitoring throughput is crucial for detection purposes.

In this research, data normalization to standardize features is utilised and scaling is done to a range typically between +1 and −1 or 0 and +1. This normalization facilitates easier comparison and evaluation of the data. For this, z-score normalization is applied to alter the original data m_i to m' using Eq. (2):

$$m' = (m_i - \mu) / \sigma \quad \ldots (2)$$

where, σ is the variance and μ is the mean of the original data.

In this investigation, two incremental online classifiers are opted for, specifically K-Nearest Neighbors and Adaptive Random Forest. The

selection of these algorithms is grounded in their widespread use in the field of online classification and their ready availability in the scikit-multiflow Python framework.

Adaptive Random Forest

ARF represents a tailored iteration designed to handle streams of data. This algorithm incorporates crucial prolonged techniques that alter the training and sampling methods of trees to better adapt to streaming data. This results in a diverse ensemble of trees. Adaptive resampling is an additional technique employed to accommodate variations in the stream by adjusting the instance weights based on their time of arrival. Essentially, more recent instances are assigned higher weights, facilitating the classifier's adaptation to concept drift over time.

K-Nearest Neighbours

KNN is a popular machine learning classification approach that relies on the distance between variables. In online classification, the algorithm uses labeled cases stored in memory to categorize newly arriving examples based on their proximity in the feature space. The algorithm uses a distance metric — typically the Euclidean distance — to calculate the distance in the data stream for the incoming data. The K nearest neighbours of m in S are then calculated based on their range. Based on the majority of its K nearest neighbours, the class label of m is assigned. The number of nearest neighbors to reflect for classification in KNN is determined by the hyper parameter K. The system can adjust its categorization decisions by informing the collection of labelled instances when new cases are acknowledged.

The proposed Adaptive Random Forest model is better than traditional machine learning algorithms such as Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB) and K-Nearest Neighbors (KNN) for detecting black hole attacks in VANET environments. ARF was able to boost the accuracy, precision, recall and F1-score due to its ensemble-based adaptive learning ability. Unlike static classifiers, ARF efficiently handled dynamic attack patterns in real-time vehicular communications and continuous data streams. KNN has a reasonable detection capability but a comparatively lower performance in scalability and adaptability. Thus, the proposed ARF-based incremental learning approach provides a more robust and efficient solution for real-time VANET intrusion detection.

Algorithm for KNN Model

Initialize KNN Model
Input: number of adjacent neighbours L, memory H
Output: KNN Model object
Initialize Model Parameters
Input: KNN Model object, L, H
Output: initialized KNN Model with parameters L and H
Calculate Euclidean Distance
Input: two instances instance1 and instance2
Output: Euclidean distance between the two instances
Classify Instance
Input: KNN Model object, new instance new_instance
Output: projected label for the new instance
Input: KNN Model object, new instance new_instance
Output: updated KNN Model with the new instance added to the memory
Process Data Stream
Input: KNN Model object, data stream data_stream
Output: KNN Model with updated predicted labels
Example Usage
Input: L, H, data_stream
Output: predicted labels for instances in the data stream
Algorithm:
Set the parameters L and H.
Initialize the KNN model.
Process the data stream.
Retrieve the predicted labels.
Display the results.

The model undergoes training on the data, and the performance is subsequently assessed on the test set to gauge its effectiveness on unseen data. Conversely, prequential assessment offers a more contemporary and alternative method for evaluating models applied to data continuously generated over time. In the prequential assessment, also known as test-then-train assessment, the fundamental concept is to initially test the model with new data instances and then train it. The experiments were conducted in a Linux-based Ubuntu environment, utilizing NS-3 as the network simulator and SUMO as the mobility simulator. The routing protocol employed was DSR, with the MAC/PHY layer configured as IEEE 802.11p Wi-Fi channel. The transmission power was set at 34 dBm, employing the User Datagram Protocol (UDP) as the transport protocol, generating Constant Bit Rate (CBR) traffic with a size of 128 bytes. The simulation involved 70 vehicles, running for a runtime of 370 s. The procedure of prequential assessment encompasses the subsequent phases, as shown in Fig. 3.

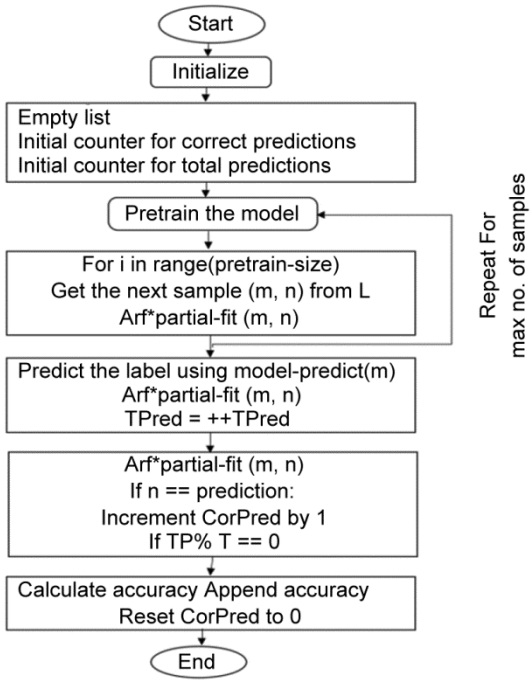


Fig. 3 — Flow chart for testing

Flowchart for Testing

Accuracy is considered as the ratio of true positives and true negatives to the total number of instances. Precision is calculated by dividing true positives by the sum of true positives and false positives. Recall is computed by dividing true positives by the sum of true positives and false negatives. Lastly, the F1-score is expressed as the harmonic mean of precision and recall, providing a balanced measure of a model's performance.

Experiments were performed in Jupyter and Python using the Matplotlib, NumPy and pandas libraries. Use of incremental online algorithms required the use of scikit-multiflow, a Python package that provides a variety of algorithms, tools and evaluation metrics for dealing with real-time data streams. The accuracy presented in the results indicates that both the KNN and ARF classifiers have accuracy of over 86% for different values of pretrain size. Specifically, the ARF classifier consistently achieves a higher level of correctness at advanced levels, achieving accuracy of 96% at the optimal ARF parameter values for the pretrain set size. Furthermore, it has been observed that both classifiers have higher accuracy as the pretrain size increases. More specifically, as depicted in Fig. 4, in terms of accuracy, ARF results are in the range of 95.63% to 93.92%, which shows a minor enhancement. However, when the pretrain size increases from 1000 to 1600, the accuracy of KNN

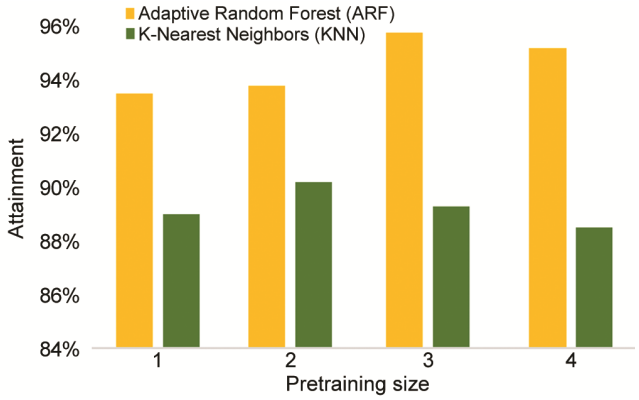


Fig. 4 — Average accuracy comparison between KNN and ARF

increases from 85.66% to 88.52%. The experimental results show that ARF successfully indicates commendable correctness with small sizes of pretraining data, while the KNN classifier shows better correctness at larger values of data size. The difference may be explained by the nature of ARF being based on ensemble learning. This feature lets ARF perform effectively with a small number of pretrained samples. The opposite is true of K-Nearest Neighbors, which works better as the size of the training data grows large. With these observations, a pretrain size of 1600 is chosen for the evaluation of the classifiers in the rest of this work.

Findings and Discourse

This section begins by examining the accuracy of the classifiers across different values, representing the count of samples utilized to train the model before initiating the prequential learning process. Subsequently, a comprehensive comparison of the classifiers is conducted based on various performance metrics, including precision, accuracy, F1-score and recall. Visual representations, in the form of plots, depict the temporal evolution of performance metrics for both classifiers. Additionally, an assessment and comparison of the running time for both classifiers are presented. The dataset used for estimation comprises 8 features and 2 classes, with 4582 samples labelled Normal and 4780 labelled black hole. The representation of performance metrics involves plotting both mean and current values for each metric.

The accuracy for the KNN and ARF algorithms is illustrated Fig. 4. Notably, ARF consistently outperforms KNN in accuracy during the evaluation period, exhibiting a mean accuracy fluctuating from 95% to 97%, while KNN shows a comparatively lower accuracy between 84% and 89%. Moreover, the

plots for mean accuracy in both KNN and ARF remain steady over the assessment period, aligning with expectations as mean accuracy provides an averaged measure of overall classifier performance across multiple test periods. This stability suggests that consistent accuracy over time is attained by both classifiers. Examining the plots of current accuracy reveals that ARF ranges from 92% to 97%, surpassing KNN, whose real-time accuracy fluctuates between 83% and 91%. In summary, the plots of current values display more variability compared to those of mean values, as the current accuracy reproduces the most recent test period's performance.

The variation of recall over time is presented in Fig. 5, monitoring both current and mean metrics. ARF results are consistently better than those of KNN, indicating the better detection capability of ARF for malicious nodes.

The corresponding comparison for precision is presented in Fig. 6. ARF again achieves better precision than KNN throughout the evaluation period, confirming that the ensemble-based incremental model produces fewer false alarms.

The overall performance comparison of the two classifiers is summarized in Fig. 7, which confirms that ARF is consistently superior to KNN across the monitored metrics.

As shown in Fig. 8, the experiments show that ARF performs better than KNN in recall, accuracy, precision and F1-score in this scenario of the study. Each classifier is unique in these respects, and the above conclusions may be attributed to that. ARF is based on the ensemble approach and has an ensemble of decision trees to improve predictive accuracy. By removing the need for an online bagging mechanism, ARF is able to make more accurate predictions and deal with data imbalance better. KNN, on the other hand, uses distance measurements and estimates based on the majority of the K nearest neighbors.

The comparison between the mean and current data for KNN and ARF is shown in Fig. 9. The next analysis is the evaluation of the training and testing time of the KNN and ARF algorithms.

The examination shows that the average training and testing time of ARF is 58.43 s, whereas the training and testing time of KNN is 51.52 s, as shown

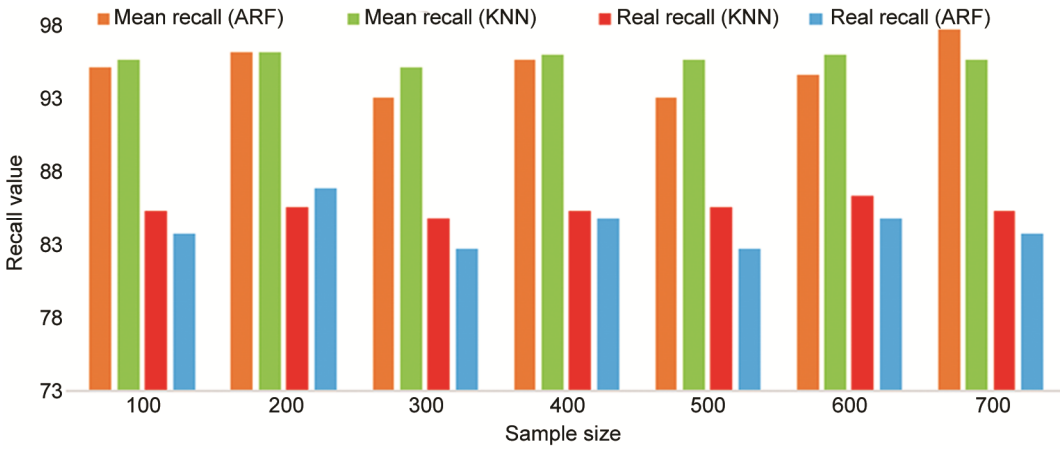


Fig. 5 — Recall value comparison for different techniques

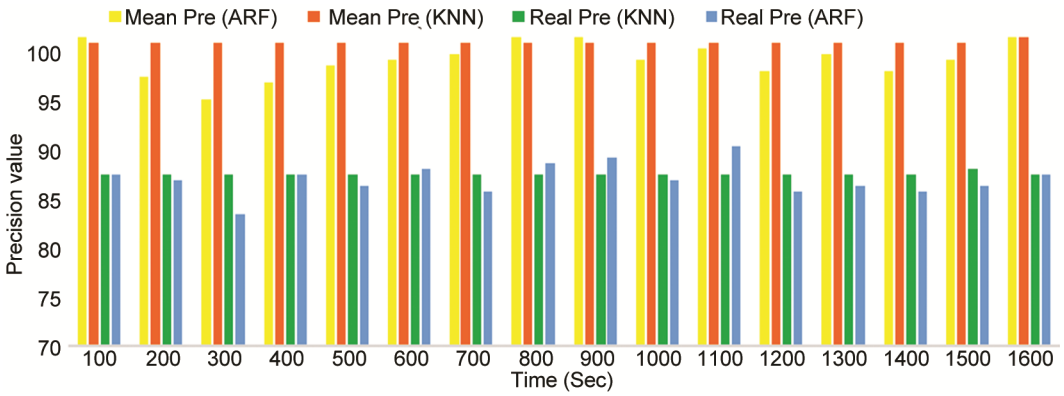


Fig. 6 — Precision comparison between KNN and ARF

in Fig. 10. The differences in time requirements of these two algorithms can be attributed to the nature of the algorithms themselves. The techniques used by KNN are relatively simple, and rely on the identification of the K nearest neighbours and conveying the most prevalent class as the label for new data. On the other hand, ARF is a set of decision trees that merges decisions from various trees, and then uses the outcome of all the trees to create the final result. Hence, the training and testing time of ARF could be longer than that of KNN due to the ensemble nature of ARF.

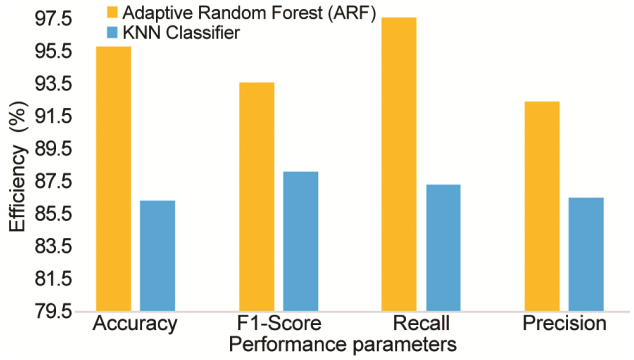


Fig. 7 — Performance comparison between KNN and ARF

Benchmarking Attack Detection: A Comparative Analysis with Leading ML Approaches

Previous studies have used SVM and Logistic Regression based continuous learning approaches for binary classification using VeReMi as the real-time dataset. Python tools are used for evaluating performance metrics like F1-score. A third, hybrid approach uses decision tree, Naïve Bayes and Random Forest techniques with a posterior finding, based on corsets, using the CICIDS2017 dataset, and tests the accuracy using Python tools.[17] Classification using ML methods (RF and KNN) is performed on the VeReMi dataset, and accuracy is used to assess their performance. In this work, classification is done using KNN and ARF classifiers on data generated employing SUMO, NS-3 and Python. The

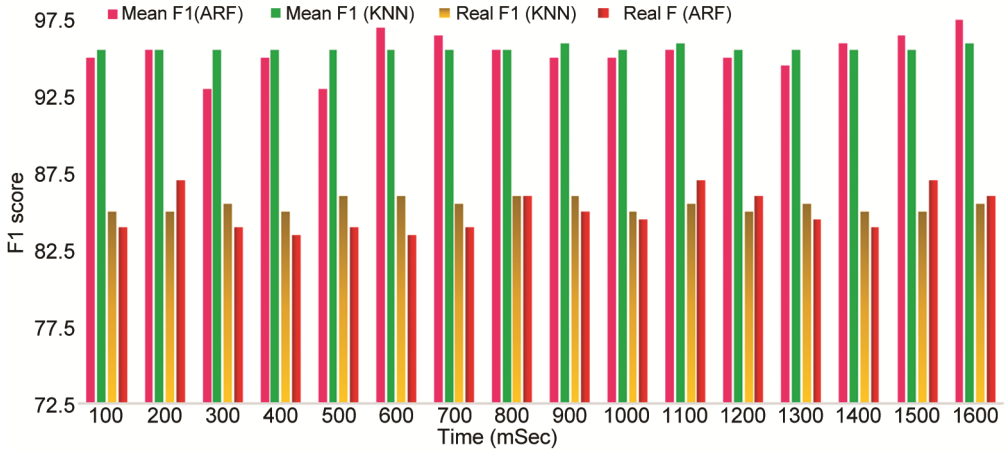


Fig. 8 — F1-score comparison between KNN and ARF for both mean and current data

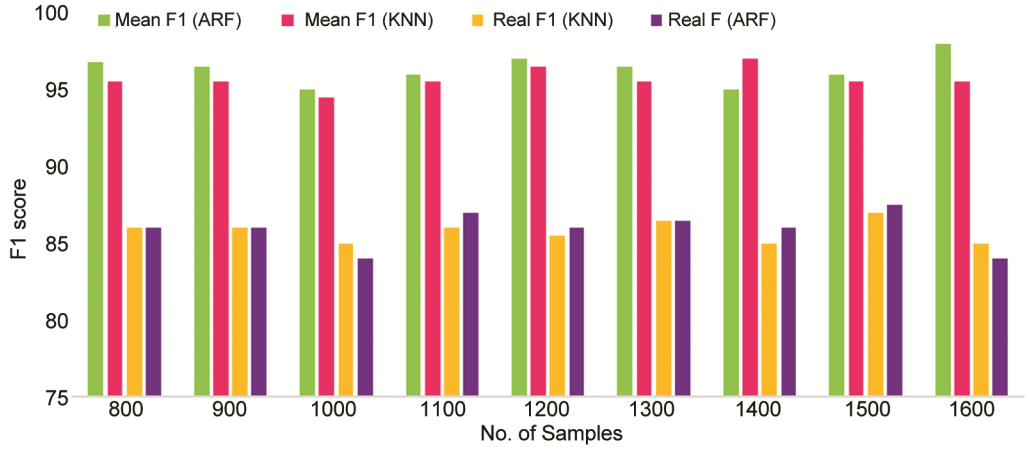


Fig. 9 — Comparison between KNN and ARF for both mean and current data

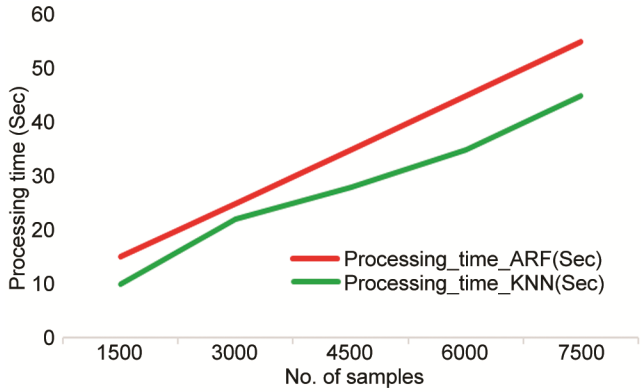


Fig. 10 — Overall processing time comparison between KNN and ARF

performance metrics include precision, recall, F1-score, and testing and training time. The study is then extended to analyze False Positive Rate (FPR), True Positive Rate (TPR) and Accuracy (ACC) metrics and study the performance of the KNN and ARF classifiers in a real-time scenario. An important benefit of our method is its ability to track the behaviour of nodes in the network in real time, which helps identify potential threats and take appropriate action in time. In addition, we build up our own dataset for attack detection, which includes important features for the detection of a black hole attack in VANETs. The simulation adopted guarantees a homogeneous VANET simulation using realistic mobility models, which are generated from a realistic map built with widely used mobility modelling tools like NS-3 and SUMO. In conclusion, the suggested method facilitates incessant learning in real time by the incremental learning process. This allows statistics to be used as they become available, and consistent models to be obtained, as opposed to a model trained on static data. This capability implies that the system can learn and enhance its predictions or decisions with the influx of new data.

Conclusions

With an emphasis on the identification of black hole attacks in the DSR protocol, this paper presents a potential method for attack detection in VANETs utilizing incremental online learning algorithms. Despite requiring longer training and testing time, the results show that ARF outperforms KNN in all important performance metrics, including precision, accuracy, F1-score and recall. The results highlight the promise of incremental learning methods for real-time, time-sensitive applications in dynamic network environments. The dynamic nature of attacks and the ongoing model adaptation to new threats, however, continue to provide difficulties. In order to improve detection accuracy while cutting down on training and testing times, future research could examine optimizing algorithm parameters. Furthermore, a hybrid strategy that combines ARF with additional online learning models may improve threat detection capabilities even more, providing a more reliable solution for VANET real-time intrusion detection.

However, the proposed model is tested with few simulation conditions and the applicability of the model in real VANET environments with different traffic and attack patterns is questionable. Despite better detection performance, the higher computational time of ARF may be challenging for large-scale deployment. Future work includes optimizing the model parameters, reducing the computational overhead, and exploring hybrid and federated learning techniques to improve real-time intrusion detection in VANETs.

References

1 Banafshehvaragh S T & Rahmani A M, Intrusion, anomaly, and attack detection in smart vehicles, *Microprocess Microsyst*, **96** (2023) 104726, doi:10.1016/j.micpro.2022.104726

2 Ajjaj S, El Houssaini S, Hain M & El Houssaini M.-A. A new multivariate approach for real time detection of routing security attacks in VANETs. *Information*, **13** (2022) 282. doi:10.3390/info13060282.

3 Chowdhury M A & Hossain M I, Machine learning in vehicular ad-hoc networks: A survey of techniques and applications, *J Comput Netw Commun*, **21** (2023) 9007641, doi:10.1155/2023/9007641.

4 Zhou Y, Xu B & Zhang X, Machine learning for security in VANETs: A review and future directions, *Wirel Netw*, **29** (2023) 2795–2815, doi:10.1007/s11276-023-03189-1.

5 Li Z & Wang L, Detection of false location data in VANETs using deep learning techniques, *IEEE Trans Veh Technol*, **73** (2024) 2801–2810, doi:10.1109/TVT.2024.3156182.

6 Singh S & Gupta A, Application of machine learning algorithms for malicious activity detection in VANETs, *Ad Hoc Netw*, **132** (2023) 102723, doi:10.1016/j.adhoc.2023.102723.

7 Tan C, Zhang L & Zhang Y, Detection of wormhole attacks in VANETs using machine learning approaches, *Comput Netw*, **205** (2023) 107508, doi:10.1016/j.comnet.2023.107508.

8 Hassan M F & Ibrahim S, Exploring SVM and KNN for wormhole attack detection in VANETs, *J Comput Sci Technol*, **38** (2023) 56–71, doi:10.1007/s11390-023-0845-4.

9 Sharma R & Kumar V, A multi-class classification approach for VANET attack detection using random forests, *Int J Wirel Mob Comput*, **26** (2024) 145–159, doi:10.1504/IJWMC.2024.10014458.

10 Patel M & Mishra R, Comparative analysis of machine learning techniques for attack detection in VANETs, *J Commun Netw*, **26** (2024) 21–36, doi:10.1109/JCN.2024.8990912.

11 Rath S & Sahu S, Advanced techniques in VANET attack detection using machine learning and multi-class classifiers, *Future Gener Comput Syst*, **135** (2023) 260–273, doi:10.1016/j.future.2023.06.030.

12 Sharma N & Gupta P, A novel approach for multi-class attack detection in VANETs using hybrid machine learning models, *J Netw Comput Appl*, **211** (2023) 102790, doi:10.1016/j.jnca.2023.102790.

13 Chen J & Xu X, Enhanced attack detection in VANETs using random forest: A multi-class classification approach, *IEEE Access*, **12** (2024) 35012–35023, doi:10.1109/ACCESS.2024.3183923.

14 Bai H & Li Z, A hybrid optimization-based deep learning model for attack detection in VANETs, *Future Internet*, **16**(1) (2024) 25, doi:10.3390/fi16010025.

15 Hassan M A & Abbas R, An intelligent intrusion detection system for VANETs using deep learning and machine learning techniques, *Comput Intell Neurosci*, (2023) 1015729, doi:10.1155/2023/1015729.

16 Zhang T & Liu H, Boosting intrusion detection system performance in VANETs using random forest and feature fusion, *Comput Commun*, **198** (2023) 98–108, doi:10.1016/j.comcom.2023.04.028.

17 Feng Z & Lee H, Detecting malevolent behavior in VANETs using supervised learning and misbehaviour datasets, *Int J Distrib Sens Netw*, (2024) 1934867, doi:10.1177/15501477241934867.

18 Ahmed S & Khan N, An efficient system for attack detection in VANETs using basic safety message approaches, *Sensors*, **23(22)** (2023) 7328, doi:10.3390/s23227328.

19 Saha P & Basak D, An efficient SVM-based approach for detecting false message attacks in VANETs, *Neural Comput Appl*, **35(11)** (2023) 8421–8432, doi:10.1007/s00542-023-07834-7.

20 Verma S & Ahuja N, A machine learning-based intrusion detection system using position features for VANETs, *J Cybersecur*, **25** (2024) 49–64, doi:10.1093/cybsec/tyz020.

21 Singh R & Kaur R, Position falsification attack detection in VANETs: A comparison of SVM and random forest approaches, *Inf Sci*, **633** (2023) 258–276, doi:10.1016/j.ins.2023.08.061.

22 Mansouri F, Ben Ahmed A & Hamdi M, A distributed intrusion detection framework for vehicular ad hoc networks using federated learning and blockchain, *Ad Hoc Netw*, **166** (2025) 103682.

23 Abdelkreem E, Elhosseini M A & Ali H A, A machine learning based scheme for enhancing the security of VANET against position spoofing attacks, *Sci Rep*, **16(1)** (2026) 1–18, doi:10.1038/s41598-025-96303-0.