Received 12 December 2023, accepted 22 December 2023, date of publication 26 December 2023,
date of current version 5 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3347619

RESEARCH ARTICLE

Enhanced Intrusion Detection in In-Vehicle Networks Using Advanced Feature Fusion and Stacking-Enriched Learning

ALI ALTALBE
Department of Computer Science, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

e-mail: Aaltalbi@kau.edu.sa

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (PSAU/2023/01/ 224252).

ABSTRACT Modern vehicles rely heavily on interconnected electronic control units (ECUs) through in-vehicle networks to perform crucial functions such as braking and monitoring engine RPMs. However, the increased number of ECUs and their connectivity to the in-vehicle network poses a security risk due to the lack of encryption and authentication protocols such as the controller area network (CAN). To address this problem, machine learning (ML) based intrusion detection systems (IDSs) have been proposed. However, existing IDSs suffer from low detection accuracy, limited real-time response, and high resource requirements. This study proposes an accurate and low-complexity IDS for in-vehicle networks based on feature fusion and ensemble learning called the Feature Fusion and Stacking-based IDS (FFS-IDS). FFS-IDS fuses multiple features extracted from raw network traffic and then classifies traffic instances into intrusive and non-intrusive categories using a stacking ensemble learning of basic machine learning classifiers. Specifically, a decision tree is employed as a base classifier, and random forest is used as a meta-learner. This work implements and validates the FFS-IDS using real-time car hacking data sets and achieves better performance than individual decision tree classifiers and popular ensemble learning methods such as Random Forest, LightGBM, AdaBoost, and ExtraTree algorithms. The results demonstrate that FFS-IDS can detect Denial of Service (DoS), Gear spoofing, and RPM spoofing attacks with up to 99% accuracy and Fuzzy attacks with up to 97.5% accuracy using benchmark datasets. Overall, this study shows the effectiveness and practicality of FFS-IDS in detecting intrusions in in-vehicle networks, which is essential for ensuring the cybersecurity and safety of modern vehicles. Future work in this area could involve exploring additional feature extraction techniques and fine-tuning hyperparameters to improve the performance of IDSs further.

INDEX TERMS Controller area network, in-vehicle network, intrusion detection system, feature fusion, ensemble learning, car hacking.

I. INTRODUCTION

Modern vehicles are equipped with electronic control units (ECUs) and robust computing systems, which have made them communication and computing-enabled terminals for intra-vehicle and inter-vehicle network communication [1], [2], [3]. This increased communication has led to more functionality and comfort, but it has also increased security threats [4], [5], [6]. The susceptibility of the controller area network (CAN) to different types of cyber attacks, including fuzzy attacks, DoS attacks, and spoofing attacks, has been a significant concern due to the lack of encryption and authentication policies in the de facto standard of in-vehicle networks [7]. Intrusion detection systems (IDSs) are vital tools for protecting in-vehicle networks by identifying unauthorized events in the networks [8], [9]. Machine learning (ML) and deep learning (DL) methods have been successfully implemented in developing effective IDSs for various applications [10], [11], [12], [13], [14].

However, most existing ML-based IDSs suffer from low detection accuracy, limited real-time response, and limited computing resources due to the availability of a large number

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

of features of network traffic [1], [15], [16], [17]. This work proposes an effective IDS (called FFS-IDS) for the CAN bus of in-vehicle networks to address these issues. FFS-IDS involves feature fusion and stacking-based ensemble learning to detect intrusions in in-vehicle networks. It fuses multiple features derived from primary features extracted from raw network traffic, capturing more information about network activity and improving the IDS's accuracy. It then classifies traffic instances into intrusive and non-intrusive categories based on stacking ensemble learning of basic ML classifiers, where a traditional decision tree is used as a base classifier and random forest is used as a meta-learner.

This work contributes to the field of in-vehicle network security by proposing a feature fusion method that combines basic features of in-vehicle network traffic to construct more comprehensive data subsets. This approach captures a broader range of information about network activity, improving the accuracy of intrusion detection. Additionally, I propose a stacking-based ensemble learning approach that further combines the outputs of multiple classifiers to improve detection performance. Specifically, we use a Bayes classifier, decision tree, and random forest classifier in a hierarchical structure to learn from the comprehensive data subsets. Finally, I validate the proposed methods using a real car hacking benchmark intrusion detection dataset for in-vehicle networks. The experimental results demonstrate that this approach significantly outperforms existing state-of-the-art methods regarding detection accuracy and false positive rate.

This paper is organized as follows. Section II provides an overview of existing research on intrusion detection in in-vehicle networks. Section III presents the intrusion detection problem in in-vehicle networks. Section IV introduces the proposed FFS-IDS system based on feature fusion and stacking-based ensemble learning for detecting intrusions in in-vehicle network traffic. The experimental setup, including the benchmark dataset and performance metrics used to evaluate the proposed system, is detailed in Section V. The results of the experiments are presented and compared with existing state-of-the-art methods in Section VI. Section VII highlights threat to validity of this work. Finally, Section VIII summarizes the contributions and discusses future directions for further research.

II. RELATED WORK

There has been a growing interest in developing effective network traffic classification methods in academia and industry in recent years. Various techniques and features have been proposed for this purpose, including deep packet inspection, port number-based classification, and statistical classification methods [11], [18], [19], [20], [21].

Deep packet inspection methods effectively identify known patterns and classify network traffic based on payload content-based information. However, they require specific hardware and have limitations in identifying multimedia-based and encrypted traffic. Port number-based classification methods use transport layer headers' port numbers to classify network traffic accurately. However, they fail to classify traffic from modern applications that do not use popular port numbers. Statistical information-based methods extract high-level features from basic packet header information, and ML and DL-based methods often use this extracted information to classify network traffic accurately. However, these methods may require significant computing resources and may not always provide high accuracy in network traffic classification [22]. These in-vehicle intrusion detection approaches' diverse strengths and limitations are compared in Table 1, allowing for a comprehensive understanding of their suitability for different scenarios.

TABLE 1. Comparison of network traffic classification methods.

Method	Effectiveness	Limitations	Resources	Accuracy
Deep packet inspection	High for known patterns	Specific hardware, multimedia/encrypted traffic issues	High	High (specific patterns)
Port number-based classification	Accurate for traditional applications	Modern applications with non-standard ports	Low	Moderate
Statistical information-based methods	Effective with ML/DL	High resources, varying accuracy	Variable	Variable

Several approaches have been proposed for detecting intrusions in in-vehicle network traffic using different techniques and features. For instance, Alshammari et al. [23] employed K nearest neighbor and support vector machine-based classifiers to detect intrusions in CAN bus traffic. Based on network traffic specifications, Olufowobi et al. [24] developed a real-time IDS for in-vehicle network traffic attacks and evaluated their system's performance using a synthetic and CAN intrusion dataset. In addition, Olufowobi et al. [25] proposed an adaptive cumulative sum method that utilizes statistical change-based information to detect attacks in CAN traffic quickly. Barletta et al. [26] used distance-based information to develop IDSs for in-vehicle networks. They suggested using the k-mean clustering algorithm with an X-Y fused Kohonen network, which demonstrated high performance in detecting intrusion from the CAN dataset. However, their system has computational complexity. Lee et al. [27] developed an IDS for detecting CAN attacks in in-vehicle network traffic using offset ratio and time interval-based information. They demonstrated the performance of their model by simulating different types of attacks, such as Fuzzy attacks, DoS attacks, and impersonation attacks.

DL methods have also been explored for detecting attacks in in-vehicle network traffic [28]. Song et al. [10] presented a deep convolutional neural network (CNN) based approach for detecting attacks in CAN traffic, which reported high attack detection accuracy. Similarly, Lo et al. [6] proposed

TABLE 2. Comparison of intrusion detection approaches in in-vehicle network traffic classification.

Category	Study	Features	Dataset	Method	Results	Strengths	Limitations
ML	Alshammari et al. [23]	Statistical features, Payload features	CAN bus traffic	KNN and SVM classifiers	High detection rate with low false positives	High accuracy in detecting intrusions in CAN bus traffic	Limited evaluation with a single dataset, may not generalize to other datasets
	Olufowobi et al. [24]	Network traffic specifications	Synthetic and CAN intrusion dataset	Real-time IDS	High detection rate with low false positives	Real-time detection of intrusions in in-vehicle network traffic	Limited evaluation using synthetic and CAN intrusion dataset, may not generalize to other datasets
Statistical-based	Olufowobi et al. [25]	Statistical change-based information	CAN traffic	Adaptive cumulative sum method	Quick detection of attacks	Fast detection of attacks in CAN traffic	Limited evaluation using synthetic and CAN intrusion dataset, may not generalize to other datasets
DL	Barletta et al. [26]	Distance-based information	CAN dataset	k-mean clustering algorithm with X-Y fused Kohonen network	High performance with computational complexity	High performance in detecting intrusions from the CAN dataset	High computational complexity
	Lee et al. [27]	Offset ratio and time interval-based information	In-vehicle network traffic	Simulation	High detection rate with low false positives	Simulated different types of attacks and demonstrated high performance of the model	Limited evaluation using a small dataset, may not generalize to other datasets
	Song et al. [10]	Deep CNN	CAN traffic	CNN	High accuracy of attack detection	High accuracy in detecting attacks in CAN traffic	Computationally expensive due to the high complexity of the CNN model
	Lo et al. [6]	Spatial and Temporal features	Car hacking dataset	CNN and LSTM	Correct classification of network traffic	Accurately classifies network traffic using spatial and temporal features	Computationally expensive due to the use of DL models
	Lokman et al. [30]	Auto-encoder	CAN traffic	Anomaly detection	Detection of anomalies in in-vehicle network	Effective in detecting anomalies in CAN traffic	Limited evaluation using a single dataset, may not generalize to other datasets
	Ashraf et al. [31]	LSTM network	UNSW-NB15 and CAN intrusion datasets	IDS	High accuracy in detecting intrusion	High accuracy in detecting intrusion in UNSW-NB15 and CAN intrusion datasets	Computationally expensive due to the use of DL models

using DL methods and different features, such as spatial and temporal features, to detect intrusions in the car hacking dataset. They used a CNN for extracting spatial features and a long short term memory (LSTM) network for extracting temporal features, and the extracted features can correctly classify network traffic of the car hacking dataset.

Leveraging the taxonomy from Table 1 and highlighting the potential of ML/DL and statistical methods, Table 2 dissects the strengths and limitations of diverse approaches, features, datasets, and results. However, direct comparisons remain challenging due to individual study goals, data sources, and evaluation criteria.

Overall, the studies discussed in this section have shown promise in detecting intrusions in in-vehicle network traffic using various methods [29]. However, each approach has its strengths and limitations. For example, DL-based methods such as CNN and LSTM networks have shown high accuracy in detecting intrusions, but they are computationally expensive due to their high complexity. On the other hand, statistical methods such as the adaptive cumulative sum method and distance-based IDS have shown promise in detecting attacks quickly with less computational complexity. Still, they may not perform as well as DL-based methods. When choosing an intrusion detection method for in-vehicle network traffic, it is essential to consider the trade-off between accuracy and computational complexity. Furthermore, more research is needed to evaluate the generalizability of these methods to different datasets and their robustness to different types of attacks. Based on the literature review presented and compared in Table 2, some research gaps can be identified:

- Limited research on statistical features: While some studies have explored statistical features, there is still a lack of research on effectively utilizing them to develop accurate and efficient IDSs for in-vehicle network traffic.
- Lack of comparative studies: Although various techniques have been proposed for detecting intrusions in

in-vehicle network traffic, there is a lack of comparative studies that evaluate and compare the performance of these techniques. Comparative studies can help identify the strengths and weaknesses of different methods and provide insights into which methods are most effective in detecting intrusions in in-vehicle network traffic.
- Limited research on resource-constrained environments: Many existing studies have focused on developing IDSs for in-vehicle network traffic in resource-rich environments without computing power or memory constraints. However, there is a lack of research on developing effective IDSs for in-vehicle network traffic in resource-constrained environments, such as those found in many embedded systems.
- Lack of focus on new types of attacks: While the existing studies have proposed different approaches for detecting various types of attacks, there is a need for more research on identifying and detecting new types of attacks that may be specific to in-vehicle network traffic. As the automotive industry continues to evolve, attackers may develop new attack techniques specific to in-vehicle network traffic, and it is crucial to have IDSs that can effectively detect such attacks.

Despite their high performance, DL models can be computationally expensive due to their high complexity. Therefore, it is crucial to develop accurate IDSs for in-vehicle network traffic that utilize less computationally expensive ML models and statistical features.

III. FORMULATING THE PROBLEM OF INTRUSION DETECTION IN IN-VEHICLE NETWORKS

Intrusion detection in in-vehicle networks identifies abnormal events or attacks in the network traffic dataset. To frame this problem, we can define the following notations:

Let DT $= i_1, i_2, \ldots, i_N$ be the set of N instances in the in-vehicle network traffic dataset, where each instance represents m-dimensional feature space I. Thus, for an instance i_j, the features are denoted as $i_j = f_{i1}, f_{i2}, f_{i3}, \ldots, f_{im}$, and $i_j \in I$.

To perform intrusion detection, we need to define a mapping ID that maps the input space I to an output space O, indicating the number of classes for network traffic classification. In binary classification, the output space consists of two classes, which can be denoted as O = intrusive, non-intrusive, normal, anomaly, 0, 1, or positive, negative. In multi-class classification, the output space has more than two classes and can be denoted as O = $o_1, o_2, \ldots, o_i$, where $i > 2$.

This work aims to find a suitable mapping ID: I $\rightarrow$ O that classifies in-vehicle network traffic into attack classes based on a given network dataset. This study proposes a decision tree-based approach, alonure fusion, and stacking methods. The proposed approach is discussed in detail in the following section.

Strengths of this problem formulation include the precise definition of notations and the focus on identifying abnormal events in in-vehicle network traffic. Limitations of this formulation include the lack of discussion on the types of attacks that can occur in in-vehicle networks and the assumption that the network dataset is already given.

IV. DESIGN OF THE PROPOSED FEATURE FUSION AND STACKING-BASED IDS (FFS-IDS)

This work proposes the Feature Fusion and Stacking based IDS (FFS-IDS) for in-vehicle networks, as shown in Figure 1. The FFS-IDS leverages multiple features extracted from raw network traffic to classify traffic instances into intrusive and non-intrusive categories using ensemble learning of basic ML classifiers in a stacking approach. The proposed system operates in three phases, which are described below.

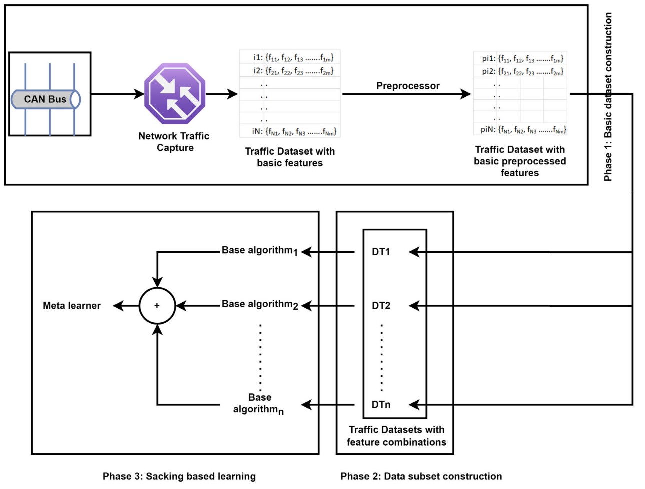
CAN Bus
Network Traffic Capture
i1: {f11, f12, f13 ....f1m}
i2: {f21, f22, f23 ....f2m}
iN: {fN1, fN2, fN3 ....fNm}
Traffic Dataset with basic features
Preprocessor
pi1: {f11, f12, f13 ....f1m}
pi2: {f21, f22, f23 ....f2m}
piN: {fN1, fN2, fN3 ....fNm}
Traffic Dataset with basic preprocessed features
Phase 1: Basic dataset construction
Meta learner
+
Base algorithm1
Base algorithm2
Base algorithmn
DT1
DT2
DTn
Traffic Datasets with feature combinations
Phase 3: Sacking based learning
Phase 2: Data subset construction

FIGURE 1. Design of the proposed feature fusion and stacking-based IDS (FFS-IDS).

A. PHASE 1 - CONSTRUCTION OF THE BASIC DATA SET

The first phase involves extracting basic features from raw network traffic and constructing a benchmark dataset. These features capture relevant information about network traffic, including spatial, temporal, and content features. Each record in the dataset represents a network traffic instance in terms of *j* features as described in Section III. The raw data may contain noise, missing, and non-uniform scale data values.

Data preprocessing is applied to prepare the captured data for further processing by ML models. This involves handling null values, removing noise, removing redundant and irrelevant information, and converting data attributes to a uniform scale.

To demonstrate the performance of the proposed FFS-IDS, we use the car-hacking dataset, which is available in CSV format and contains fields such as Timestamp, ID, DLC, D0-D7, and Tag. Table 3 describes the attributes of the car-hacking dataset.

The car-hacking dataset contains data fields of various types, including time, categorical, and numeric fields. However, this raw data cannot be directly used for ML

TABLE 3. Attributes of the car-hacking dataset.

Attribute	Description
Timestamp	The recorded time of attack message
ID	Identifier of CAN message in Hex
DLC	The number of data bytes sent on the network varies from 0 to 8
D0-D7	Data bytes sent over the network
Tag	Tag of the message as normal (R) or attack (T)

purposes. A CAN traffic preprocessor module is used to make it compatible with ML algorithms that require numeric input, as shown in Figure 2.

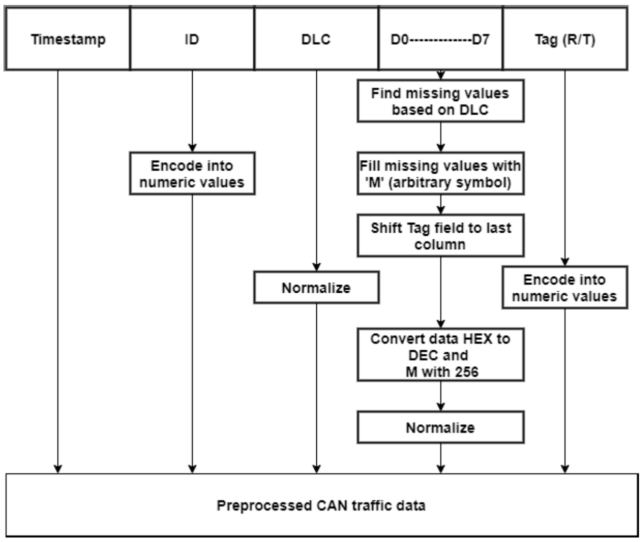
Timestamp
ID
DLC
D0-------------D7
Tag (R/T)
Find missing values based on DLC
Fill missing values with 'M' (arbitrary symbol)
Shift Tag field to last column
Encode into numeric values
Normalize
Encode into numeric values
Convert data HEX to DEC and M with 256
Normalize
Preprocessed CAN traffic data

FIGURE 2. CAN traffic preprocessor for in-vehicle network intrusion detection.

The preprocessing step involves encoding the ID field, which represents the unique code for each message sent on the CAN bus, into a numeric value ranging from 0 to the maximum number of IDs (Max IDs). In the car-hacking dataset, there are 29 unique CAN message IDs. The DLC field, indicating the number of bytes sent over the network, is normalized to a range of 0 to 1 using Eqs. 1 and 2 [32]. Here, val_i is the initial feature value i, while Min_i and Max_i are the minimum and maximum values of feature i, respectively [33].

$$Normalizedvalue_i = normalize(ln(val_i + 1)) \tag{1}$$

$$normalize(x_i) = \frac{x_i - ln(Min_i + 1)}{ln(Max_i + 1) - ln(Min_i + 1)} \tag{2}$$

Here, val_i is the initial feature value i, while Min_i and Max_i are the minimum and maximum values of feature i, respectively [33].

The data fields (D0 to D7) contain data bytes in HEX format, and the DLC field value indicates the length of the data fields. The preprocessor module shifts the tag field to the last column and fills non-available data bytes with an arbitrary symbol, 'M'. All values of D0 to D7 are converted from HEX format to DEC format, and 'M' is replaced with 256. Finally, all values of D0 to D7 are normalized to a range of 0 to 1 using Eqs. 1 and 2.

To encode the tag field into numeric values, Normal (R) is converted to 0, and Attack (T) is converted to 1 for further processing with neural networks. The sklearn python library was used to perform the data cleaning methods mentioned above, specifically, the sklearn.preprocessing.LabelEncoder function for encoding categorical to numeric values and the sklearn.preprocessing.normalize function for normalizing values to a uniform scale.

B. PHASE 2 - FEATURE COMBINATION-BASED DATA SUBSET CONSTRUCTION

In Phase 2, we aim to construct a subset of the dataset by combining multiple features generated in Phase 1. As different features have different capabilities to detect anomalous behaviour, using a relevant feature set is crucial in the ML pipeline. I propose constructing comprehensive data subsets based on feature fusion concepts to address this issue.

Network traffic data contain different types of information analyzed in different dimensions, namely spatial and temporal aspects of network data and data content regarding network data behaviour. Efficient network traffic classification requires temporal, spatial, and content features derived from basic network traffic features, supporting and complementing each other in detecting anomalies. Hence, combining different features can result in accurate network traffic classification by characterizing different anomalies using comprehensive data features. Using a fixed or single-feature dataset may not suffice in detecting anomalies in a complex network such as the in-vehicle network.

I propose constructing a comprehensive data subset by taking all permutations among different features using a feature fusion method to address this issue. This approach ensures both accuracy and diversity in network traffic by combining different feature sets.

C. PHASE 3 - STACKING-BASED ENSEMBLE LEARNING

In Phase 3, we utilize a stacking-based ensemble learning approach that combines the outputs of base classifiers trained on each subset of the dataset generated in Phase 2. While each basic algorithm is trained using a comprehensive feature data subset for partial network traffic learning, it only predicts the probability of a specific network traffic class. The goal of the meta-learner is to take the output of the basic algorithms and produce an overall detection of the network traffic class that is more generalized and comprehensive.

To achieve this goal, we first train a set of basic learning algorithms $(BA_1, BA_2, \ldots, BA_n)$ to create basic models $(BM_1, BM_2, \ldots, BM_n)$ using comprehensive feature data subsets $(DT_1, DT_2, \ldots, DT_n)$. The predicted probabilities of these basic models, $BA_1(p_1, p_2, \ldots, p_n), BA_2(p_1, p_2, \ldots, p_n), \ldots, BA_n(p_1, p_2, \ldots, p_n)$ are then fed to the meta learner. Based on a two-level stacking approach, the final model (FM) is trained using ensemble learning of basic models $(BM_1, BM_2, \ldots, BM_n)$.

The proposed feature fusion and stacking-based IDS algorithm is presented in Algorithm 1. The computational cost of this algorithm depends on updating FM and is $O(N * \beta)$, where β is a constant value less than N. The feature combination-based data subset construction phase requires $O(1)$. Therefore, the overall computational complexity of the proposed algorithm in terms of space and time is $O(N)$.

Algorithm 1 FFS-IDS Algorithm

```
Require: $DT_i$: $i^{th}$ Basic data subset, BA: Basic learning
    algorithm, BM: Basic learning model, ML: Meta learning
    algorithm
Ensure: Network traffic class predictions of the final
    meta-learning model (FM)
 1: Extract basic features from raw network traffic data
 2: Construct comprehensive data subsets from basic fea-
    tures using a combination and permutation approach
 3: Set $i \leftarrow 1$.
 4: while $i \leq N$ do
      $BM_i$ = BA ($DT_i$)
 5: end while
 6: $DT' \leftarrow$ NULL
 7: Set $i \leftarrow 1$.
 8: while $i \leq N$ do
 9:   Set $j \leftarrow 1$.
10:   while $j \leq N$ do
11:     $EN_{ij} = BM_j\ (DT_j)$
12:     $DT' += $ y $(EN_{ij})$
13:   end while
14: end while
15: FM $\leftarrow$ y $(DT')$
16: Return FM
```

The proposed system's overall effectiveness and accuracy depend significantly on the accuracy of the base models. To ensure high performance and computational efficiency, I carefully selected the decision tree algorithm as the base learning algorithm and the random forest algorithm as the meta-learning algorithm. The decision tree algorithm is well-suited for classification tasks, providing highly accurate results with minimal computations [34], [35]. On the other hand, the random forest algorithm is known to achieve high classification accuracy through multiple decision trees, even in the presence of noise and overfitting issues [35], [36].

V. EXPERIMENTAL SETUP AND IMPLEMENTATION

This section presents a comprehensive overview of the experimental setup, implementation, dataset, performance metrics, and results of the proposed approach for detecting intrusions in in-vehicle networks. I also provide a detailed analysis of the results and highlight significant observations from the comparative analysis with the outcomes of existing approaches.

A. EXPERIMENTAL SETUP

To implement the proposed FFS-IDS, and state of the art methods including DT, RF, LightGBM and ExtraTree methods, I utilized the Anaconda distribution of Python and various libraries such as Pandas, Numpy, and Scikit-learn for loading the dataset, performing pre-processing operations, constructing a comprehensive feature combination-based data subset, and using decision tree and random forest algorithms as base learning and meta-learning algorithms, respectively. The experiments were conducted on a machine with an Intel Core I3-2330M CPU @ 2.20 GHz, 4 GB RAM, and 1 TB HDD running on the Windows operating system. The results are recorded for FFS-IDS and the identified algorithms, DT, RF, LightGBM and ExtraTree methods.

To ensure fair comparisons, we used the default hyper-parameters of the identified algorithms as defined in the Scikit-learn library, as presented in Tables 4 - 8. I also utilized commonly used performance metrics to evaluate the effectiveness of the proposed FFS-IDS approach. I compared the results with existing approaches for detecting intrusions in in-vehicle networks. Furthermore, I analyzed the results and highlighted significant observations from the comparative analysis.

TABLE 4. Hyper-parameters of decision Tree classifier.

Parameter	Description	Default Value
criterion	Splitting criterion	"gini" (Gini impurity)
splitter	Split strategy	"best" (chooses best split)
max_depth	Maximum depth of tree	None (no limit)
min_samples_leaf	Minimum number of samples per leaf	1
min_samples_split	Minimum number of samples required for a split	2
max_features	Number of features to consider at each split	"auto" (all features)
random_state	Seed for RNG	None
verbose	Logging level	0 (no logging)

TABLE 5. Hyper-parameters of random forest classifier.

Parameter	Description	Default Value
n_estimators	Number of trees in the forest	100
max_depth	Maximum depth of a tree	None (no limit)
min_samples_split	Minimum number of samples required for split	2
min_samples_leaf	Minimum number of samples required in a leaf	1
bootstrap	Whether to use bootstrap sampling	True
max_features	Number of features to consider at each split	'auto' (sqrt of total features)
oob_score	Whether to compute out-of-bag scores	False
random_state	Seed for the random number generator	None
verbose	Logging level	0

TABLE 6. Hyper-parameters of LightGBM classifier.

Parameter	Description	Default Value
Boosting Type	Boosting algorithm	gbdt
Objective	Loss function	binary (for binary classification)
n_estimators	Number of boosting rounds	100
learning_rate	Learning rate	0.1
num_leaves	Number of leaves per tree	31
feature_fraction	Fraction of features to consider	0.9
bagging_fraction	Fraction of data points for bagging	0.8
bagging_freq	Bagging frequency	5
min_child_samples	Minimum child samples	20
min_split_gain	Minimum gain required for split	0.0
max_depth	Maximum depth of the tree	-1 (no limit)
random_state	Seed for RNG	None
verbose	Logging level	-1 (no logging)

TABLE 7. Hyper-parameters of AdaBoost classifier.

Parameter	Description	Default Value
base_estimator	Weak learner type	None
n_estimators	Number of boosting rounds	50
learning_rate	Shrinkage parameter	1.0
algorithm	Boosting algorithm variant	'SAMME.R'
random_state	Seed for RNG	None
verbose	Logging level	0

TABLE 8. Hyper-parameters of ExtraTree classifier.

Parameter	Description	Default Value
n_estimators	Number of trees in the forest	100
max_depth	Maximum depth of each tree	None (no limit)
min_samples_split	Minimum number of samples required to split a node	2
min_samples_leaf	Minimum number of samples required at each leaf	1
bootstrap	Whether to bootstrap the data when building trees	True
max_features	Number of features to consider at each split	"auto" (sqrt of total features)
oob_score	Whether to compute out-of-bag score	False
random_state	Seed for the random number generator	None
verbose	Logging level	0 (no logging)
class_weight	Weights associated with classes	None
warm_start	Use warm starting when fitting	False

B. BENCHMARK DATASET

To assess the performance of the proposed FFS-IDS system in detecting intrusions in in-vehicle networks, we utilized the car hacking dataset introduced in [37]. This dataset comprises in-vehicle network traffic data recorded by ECUs over the CAN bus and includes normal and attack traffic. The dataset comprises messages transmitted by ECUs using specific identifiers, which are then received by all connected ECUs [10], [38].

The car hacking dataset includes attacks that disrupt normal vehicle operations, such as braking and RPM gauges. It comprises five types of attacks: DoS attacks, fuzzy attacks, and spoofing attacks on the gear system and RPM gauge, in addition to normal data instances over the CAN bus.

The car hacking dataset lists its features, as presented in Table 3. The dataset was created by logging in-vehicle network traffic through a real vehicle's on-board diagnosis (OBD-II) port. Attack traffic was injected by adding fabricated messages to the in-vehicle network. The dataset comprises 300 instructions for each respective attack class, with each instruction lasting for 3 to 5 seconds. The collected data is presented in CSV format, with separate files for normal and attack traffic, DoS, gear spoofing, RPM spoofing, and fuzzy attacks. Table 9 details the data instances used to validate the proposed FFS-IDS system.

TABLE 9. Car-hacking dataset.

Dataset	Normal	Attack
Normal instances	988872	NA
DoS data subset instances	3078250	587521
Fuzzy data subset instances	3347013	491847
Gear Spoofing data subset instances	3845890	597252
RPM Spoofing data subset instances	3966805	654897

I divided the car hacking dataset into training and testing datasets in the ratio shown in Table 10 for training and testing. The experimental setup utilized Python programming language, the Anaconda distribution, and libraries such as Pandas, numpy, and sklearn for dataset loading, preprocessing operations, and constructing a comprehensive feature combination-based data subset.

TABLE 10. Training and test datasets.

Dataset	Type	Training	Test
DoS data subset instances	Attack	393964	193557
	Normal	2062102	1016148
Fuzzy data subset instances	Attack	329502	162345
	Normal	2242534	1104479
Gear Spoofing data subset instances	Attack	400719	196533
	Normal	2576186	1269704
RPM Spoofing data subset instances	Attack	438245	216652
	Normal	2658295	1308510

C. PERFORMANCE METRICS

To analyze and compare the performance of the proposed FFS-IDS system and existing ML approaches for detecting intrusions in-vehicle networks, I computed commonly used performance metrics, including classification accuracy, false positive rate, and true positive rate. These metrics are typically computed from the confusion matrix, which represents the classification results of the IDS. The elements

TABLE 11. Confusion matrix elements.

Sr No.	Element	Definition
1.	True Negative (TN)	Normal/non-intrusive behavior that is successfully labeled as normal/non-intrusive by the IDS.
2.	True positive (TP)	Intrusions that are successfully detected by the IDS.
3.	False positive (FP)	Normal/non-intrusive behavior that is wrongly classified as intrusive by the IDS.
4.	False Negative (FN)	Intrusions that are missed by the IDS, and classified as normal/non-intrusive.

TABLE 12. Confusion matrix.

Actual	Predicted	
	Normal	Attack
Normal	TN	FP
Attack	FN	TP

of the confusion matrix are defined in Table 11. The possible outcomes for classifying events are shown in Table 12.

While the confusion matrix is a powerful tool for representing the classification results of IDSs, it may not be beneficial for comparing different IDSs. Various performance metrics have been defined in terms of the confusion matrix variables to address this issue. These metrics produce numerical values that can be easily compared, providing insight into the overall performance of the IDS. Some commonly used performance metrics include classification accuracy, false-positive rate, and true-positive rate [39], [40], [41], [42]. By evaluating these metrics, I can analyze and compare the effectiveness of the proposed FFS-IDS system with existing ML approaches for detecting intrusions in in-vehicle networks.

1) Classification accuracy: It is defined as the ratio of correctly classified instances and the total number of instances.

$$CR = \frac{Correctly\_classified\_instances}{Total\_number\_of\_instances} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

2) Detection rate or Recall: It is computed as the ratio between the number of correctly detected attacks and the total number of attacks.

$$DR = \frac{Correctly\_detected\_attacks}{Total\_number\_of\_attacks} = \frac{TP}{TP + FN} \tag{4}$$

3) False positive rate (FPR): It is defined as the ratio between the number of normal instances detected as attack and the total number of normal instances.

$$FPR = \frac{Number\_of\_normal\_instances\_detected\_as\_attacks}{Total\_number\_of\_normal\_instances} = \frac{FP}{FP + TN} \tag{5}$$

4) Precision (PR): It is the fraction of data instances predicted as positive that are actually positive.

$$PR = \frac{TP}{TP + FP} \tag{6}$$

5) F-measure (FM): For a given threshold, the FM is the harmonic mean of the precision and recall at that threshold.

$$FM = \frac{2}{\frac{1}{PR} + \frac{1}{Recall}} \tag{7}$$

VI. RESULTS AND DISCUSSION

This study compares the performance of the proposed FFS-IDS system with other commonly used classifiers, namely the decision tree [43], random forest [44], LightGBM [45], AdaBoost [46], and ExtraTree [47], which are ensemble learning methods. The evaluation of these methods is based on the car hacking dataset.

I conducted ten independent experiments using FFS-IDS and the other classifiers with their default hyperparameters.

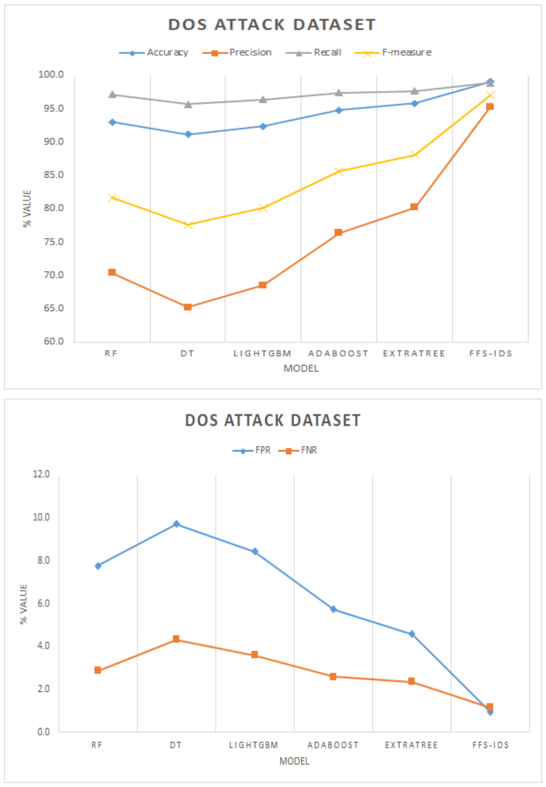

FIGURE 3. Comparison of the detection performance of the ffs-ids system and other state-of-the-art methods on the DoS attack dataset.

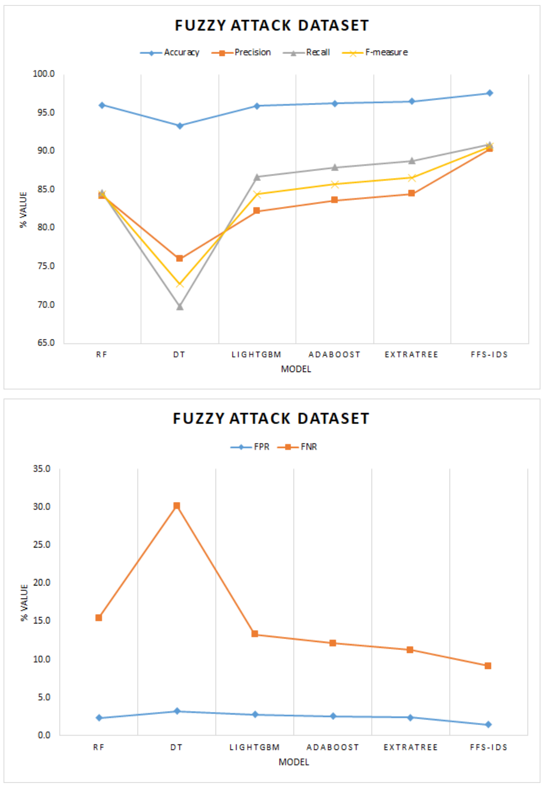

FIGURE 4. Comparison of the detection performance of the FFS-IDS system and other state-of-the-art methods on the fuzzy attack dataset.

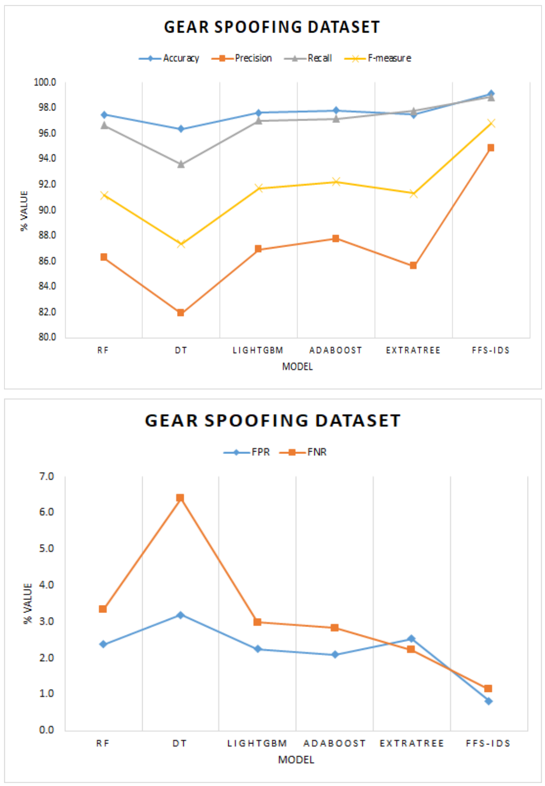

FIGURE 5. Comparison of the detection performance of the FFS-IDS system and other state-of-the-art methods on the gear spoofing attack dataset.

TABLE 13. Comparison of intrusion detection methods for dos attacks on in-vehicle networks.

Method	Accuracy	Precision	Recall	F-measure	FPR	FNR
RF	93.0106	70.4131	97.1306	81.6416	7.7742	2.8694
DT	91.1438	65.2186	95.6726	77.5634	9.7189	4.3274
LightGBM	92.3427	68.5362	96.3969	80.1134	8.4296	3.6031
AdaBoost	94.7714	76.4057	97.3987	85.6344	5.7291	2.6013
ExtraTree	95.7719	80.2230	97.6472	88.0817	4.5854	2.3528
FFS-IDS	99.0156	95.1942	98.8376	96.9817	0.9505	1.1624

TABLE 14. Comparison of intrusion detection methods for Fuzzy attacks on in-vehicle networks.

Method	Accuracy	Precision	Recall	F-measure	FPR	FNR
RF	95.9915	84.2108	84.5785	84.3942	2.3310	15.4215
DT	93.3077	75.9962	69.8358	72.7859	3.2423	30.1642
LightGBM	95.8887	82.2031	86.6858	84.3849	2.7586	13.3142
AdaBoost	96.2368	83.5852	87.8961	85.6865	2.5372	12.1039
ExtraTree	96.4661	84.4552	88.7616	86.5549	2.4014	11.2384
FFS-IDS	97.5735	90.2819	90.8436	90.5619	1.4373	9.1564

TABLE 15. Comparison of intrusion detection methods for gear spoofing attacks on in-vehicle networks.

Method	Accuracy	Precision	Recall	F-measure	FPR	FNR
RF	97.4898	86.2737	96.6494	91.1673	2.3802	3.3506
DT	96.3749	81.9309	93.5965	87.3761	3.1951	6.4035
LightGBM	97.6470	86.9407	97.0188	91.7037	2.2557	2.9812
AdaBoost	97.8024	87.7506	97.1689	92.2199	2.0995	2.8311
ExtraTree	97.5038	85.6352	97.7790	91.3051	2.5388	2.2210
FFS-IDS	99.1287	94.8628	98.8531	96.8169	0.8286	1.1469

The performance of these classifiers was evaluated using commonly used performance metrics. To compare the results of these experiments, I visually represented the experimental results using Figure 3 – 6.

Tables 13 – 16 summarize the comparative analysis of the proposed FFS-IDS system with the existing approaches for detecting intrusion in in-vehicle networks based upon car hacking data set regarding the accuracy, precision, recall, f measure, false-positive rate and false-negative rate.

Figures 3 – 6 and Tables 13 – 16 demonstrate that FFS-IDS outperforms the baseline methods in detecting intrusions from the car hacking dataset, achieving higher accuracy, precision, recall, F-measure, FPR, and FNR. FFS-IDS performs better in detecting DoS and spoofing attacks than fuzzy attacks, which exhibit more complex behaviour.

Specifically, FFS-IDS achieved detection rates of up to 99% for DoS, gear spoofing, and RPM spoofing attacks, and up to 97.5% for fuzzy attacks, with a significantly reduced FPR of 0.95% for DoS attacks compared to the other individual and ensemble learning methods. The precision, recall, F-measure, and FNR metrics also show similar superior performance for the DoS attack class, as reported in Table 13.

The comparative results presented in Figures 3 – 6 further validate the effectiveness of FFS-IDS in detecting various

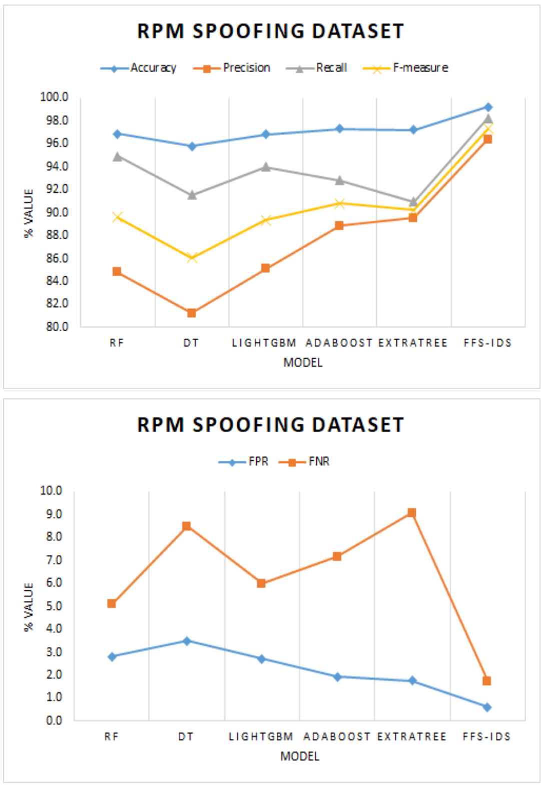

FIGURE 6. Comparison of the detection performance of the FFS-IDS system and other state-of-the-art methods on the RPM spoofing attack dataset.

TABLE 16. Comparison of intrusion detection methods for RPM spoofing attacks on in-vehicle networks.

Method	Accuracy	Precision	Recall	F-measure	FPR	FNR
RF	96.8622	84.8024	94.9223	89.5774	2.8166	5.0777
DT	95.7896	81.2167	91.5283	86.0648	3.5048	8.4717
LightGBM	96.8102	85.1031	93.9987	89.3300	2.7243	6.0013
AdaBoost	97.3244	88.8393	92.8263	90.7890	1.9308	7.1737
ExtraTree	97.2057	89.5561	90.9338	90.2397	1.7558	9.0662
FFS-IDS	99.2207	96.3853	98.2522	97.3098	0.6171	1.7478

attack classes on different datasets, indicating that the feature fusion-based subset of the car hacking dataset, integrated with a stacking-based ensemble learning method, can improve the performance of IDS significantly over the individual decision tree classifier and most popular ensemble learning methods. This dataset's feature construction, followed by the stacking-based ensemble learning method, extracts helpful information for classifying normal and attack network traffic.

Traditional individual classifiers and popular ensemble learning methods reported less accurate results with high FPR and FNR values than FFS-IDS, mainly due to their inability to extract relevant information for normal and attack traffic classification. Moreover, these methods reported poor performance in detecting the fuzzy attack class due to the complex behaviour of fuzzy attacks based on injected messages. Fuzzy attacks are difficult to detect compared to other attack classes such as spoofing and DoS attacks,

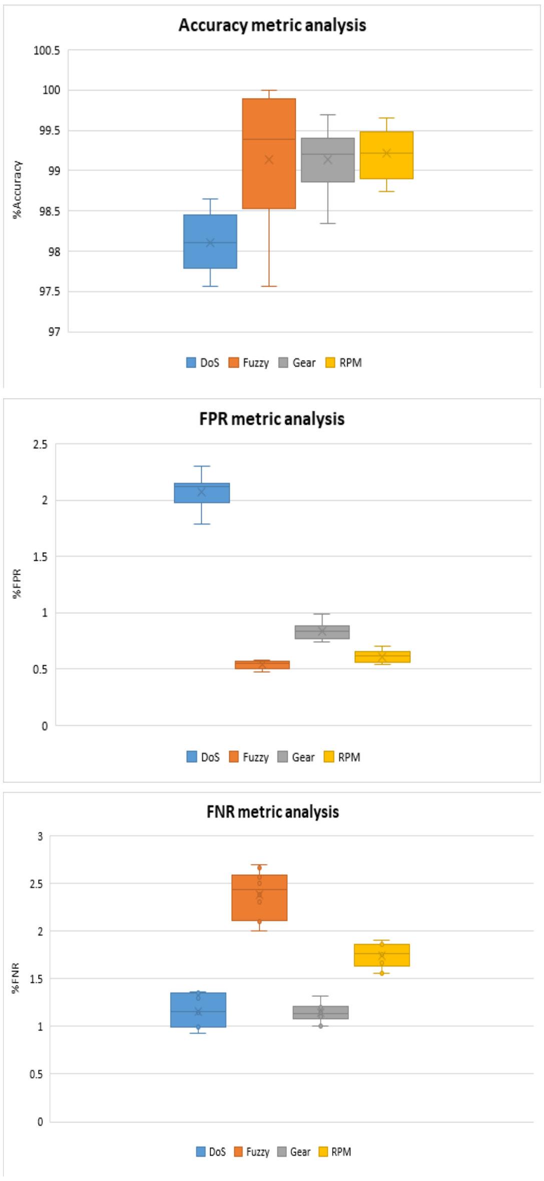

FIGURE 7. Box plot-based analysis over 10 independent experiments.

which require regular injection of attack messages into the in-vehicle network. Regular injection of attack messages can be easily detected for spoofing and DoS attacks. However, fuzzy attack messages are injected into the network less frequently.

Figure 7 shows box plots of the accuracy, FPR, and FNR metrics. It can be observed that the proposed FFS-IDS has reported stable results in detecting different attack classes, except for the fuzzy attack class, due to its complex behaviour.

VII. ADDRESSING POTENTIAL THREATS TO VALIDITY

Transparency and robustness are paramount in this research, and I comprehensively address potential threats to the validity of this study across four dimensions: internal, external, construct and conclusion validity.

- **Threats to External Validity:** The generalization of these findings may be limited due to using a specific car-hacking dataset. While this dataset captures various in-vehicle network scenarios, the diversity in real-world conditions may not be fully represented. Additionally, the specific characteristics of the attacks in the dataset may not cover the entire spectrum of potential intrusions in in-vehicle networks.
- **Threats to Internal Validity:** The experimental design involves using default hyperparameters for machine learning classifiers, which could influence the internal validity as the chosen parameters may not be optimal for the specific characteristics of the dataset. Moreover, the proposed FFS-IDS algorithm's performance is evaluated based on a specific configuration, and changes in the dataset or algorithmic parameters might impact the results.
- **Threats to Construct Validity:** The feature extraction techniques employed in this study focus on specific aspects of network traffic. Variations in network architectures or the introduction of new attack methodologies might threaten the construct validity, as the chosen features may not comprehensively cover all potential intrusions.
- **Threats to Conclusion Validity:** The conclusions drawn from the results are based on the specific dataset, experimental setup, and evaluation metrics chosen. Changes in any of these elements or introducing new metrics could potentially alter the conclusions drawn from this study.

VIII. CONCLUSION AND FUTURE WORK

The increasing number of ECUs in modern vehicles has led to an increasingly connected internal network, the CAN, which has made them vulnerable to malicious attacks. This work proposed an effective IDS for in-vehicle networks called FFS-IDS, which uses feature fusion and stacking-based ensemble learning. FFS-IDS fuses multiple features extracted from raw network traffic and classifies traffic instances into intrusive and non-intrusive categories based on stacking ensemble learning of basic ML classifiers.

The experimental results demonstrated that FFS-IDS outperformed state-of-the-art IDSs in terms of detection performance, achieving detection accuracies of up to 99% for DoS, Gear spoofing, and RPM spoofing attacks, and up to 97.5% for Fuzzy attacks on the car hacking benchmark dataset. This research demonstrates the effectiveness and practicality of FFS-IDS for detecting intrusions in in-vehicle networks.

The future work outlined in the paper encompasses addressing identified limitations and enhancing the proposed FFS-IDS for in-vehicle networks. The paper acknowledges the constraints of using a single dataset for evaluation and default hyperparameters for machine learning classifiers. To overcome these limitations, additional feature extraction techniques can be explored to enhance the detection performance of IDSs. Furthermore, the intention is to fine-tune the hyperparameters of base algorithms, ensuring a more robust and accurate IDS.

The future research directions involve empirical validation and optimization of the FFS-IDS algorithm. The authors propose conducting thorough experiments to measure the execution time on diverse hardware configurations, analyzing each algorithmic stage's time consumption. Scalability concerning dataset size and complexity will be rigorously assessed. Additionally, a detailed analysis of resource consumption will be conducted, including memory usage, CPU and GPU utilization, and network bandwidth requirements. The authors aim to compare the resource consumption of their approach with existing IDS solutions, evaluating its feasibility for real-world deployment.

The optimization focus includes exploring various techniques to enhance the efficiency of the FFS-IDS algorithm. This involves investigating alternative feature fusion methods, optimizing data subset construction algorithms, and exploring lightweight stacking classifier architectures. The overarching goal is to reduce execution time and resource consumption while maintaining or improving the detection accuracy of the system.

Moreover, the paper proposes investigating hardware-specific adaptations, tailoring the FFS-IDS algorithm for platforms like embedded devices or edge computing environments. This involves developing specialized implementations that leverage the strengths of available hardware resources while minimizing constraints.

REFERENCES

[1] Z. Bi, G. Xu, G. Xu, M. Tian, R. Jiang, and S. Zhang, "Intrusion detection method for in-vehicle CAN bus based on message and time transfer matrix," *Secur. Commun. Netw.*, vol. 2022, pp. 1–19, Mar. 2022.

[2] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—Architectures, enabling technologies, applications, and development areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018.

[3] W. Jo, S. Kim, H. Kim, Y. Shin, and T. Shon, "Automatic whitelist generation system for Ethernet based in-vehicle network," *Comput. Ind.*, vol. 142, Nov. 2022, Art. no. 103735.

[4] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Netw.*, vol. 31, no. 5, pp. 50–58, 2017.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Secur. Symp. (USENIX Secur.)*, 2011, pp. 1–16.

[6] W. Lo, H. Alqahtani, K. Thakur, A. Almadhor, S. Chander, and G. Kumar, "A hybrid deep learning based intrusion detection system using spatial–temporal representation of in-vehicle network traffic," *Veh. Commun.*, vol. 35, Jun. 2022, Art. no. 100471.

[7] L. Yang and A. Shami, "A transfer learning and optimized CNN based intrusion detection system for Internet of Vehicles," 2022, *arXiv:2201.11812*.

[8] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "An effective in-vehicle CAN bus intrusion detection system using CNN deep learning approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[9] M. Alolaiwy, M. Tanik, and L. Jololian, "From CNNs to adaptive filter design for digital image denoising using reinforcement Q-learning," in *Proc. SoutheastCon*, Mar. 2021, pp. 1–8.

[10] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.

[11] L. Yang, A. Moubayed, A. Shami, P. Heidari, A. Boukhtouta, A. Larabi, R. Brunner, S. Preda, and D. Migault, "Multi-perspective content delivery networks security framework using optimized unsupervised anomaly detection," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 686–705, Mar. 2022.

[12] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: Recent developments and challenges," *Soft Comput.*, vol. 25, no. 15, pp. 9731–9763, Aug. 2021.

[13] R. Kaur, M. Sachdeva, and G. Kumar, "Nature inspired feature selection approach for effective intrusion detection," *Indian J. Sci. Technol.*, vol. 9, no. 42, pp. 1–9, Nov. 2016.

[14] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, G. Madzudzo, and M. Cheah, "AI-based intrusion detection systems for in-vehicle networks: A survey," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–40, Nov. 2023.

[15] A. K. Desta, S. Ohira, I. Arai, and K. Fujikawa, "Rec-CNN: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots," *Veh. Commun.*, vol. 35, Jun. 2022, Art. no. 100470.

[16] S. T. Banafshehvaragh and A. M. Rahmani, "Intrusion, anomaly, and attack detection in smart vehicles," *Microprocessors Microsystems*, vol. 96, Feb. 2023, Art. no. 104726.

[17] Y. Xun, Z. Deng, J. Liu, and Y. Zhao, "Side channel analysis: A novel intrusion detection system based on vehicle voltage signals," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 7240–7250, Jun. 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10016748, doi: 10.1109/TVT.2023.3236820.

[18] Y. D. Goli and R. Ambika, "Network traffic classification techniques—A review," in *Proc. Int. Conf. Comput. Techn., Electron. Mech. Syst. (CTEMS)*, Dec. 2018, pp. 219–222.

[19] R. Kaur, M. Sachdeva, and G. Kumar, "Study and comparison of feature selection approaches for intrusion detection," in *Proc. IJCA Int. Conf. Adv. Emerging Technol. (ICAET)*, no. 2, Sep. 2016, pp. 1–7.

[20] K. K. Sheena and G. Kumar, "Analysis of feature selection techniques: A data mining approach," in *Proc. Int. Conf. Adv. Emerg. Technol.*, 2016, pp. 17–21.

[21] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4738466, doi: 10.1109/SURV.2008.080406.

[22] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 2451–2455.

[23] A. Alshammari, M. A. Zohdy, D. Debnath, and G. Corser, "Classification approach for intrusion detection in vehicle systems," *Wireless Eng. Technol.*, vol. 9, no. 4, pp. 79–94, 2018.

[24] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "SAIDuCANT: Specification-based automotive intrusion detection using controller area network (CAN) timing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1484–1494, Feb. 2020.

[25] H. Olufowobi, U. Ezeobi, E. Muhati, G. Robinson, C. Young, J. Zambreno, and G. Bloom, "Anomaly detection approach using adaptive cumulative sum algorithm for controller area network," in *Proc. ACM Workshop Automot. Cybersecurity*, Mar. 2019, pp. 25–30.

[26] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, "A Kohonen SOM architecture for intrusion detection on in-vehicle communication networks," *Appl. Sci.*, vol. 10, no. 15, p. 5062, Jul. 2020.

[27] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 57–5709.

[28] J. Xiao, L. Yang, F. Zhong, H. Chen, and X. Li, "Robust anomaly-based intrusion detection system for in-vehicle network by graph neural network framework," *Int. J. Speech Technol.*, vol. 53, no. 3, pp. 3183–3206, Feb. 2023.

[29] H. Alqahtani and G. Kumar, "A deep learning-based intrusion detection system for in-vehicle networks," *Comput. Electr. Eng.*, vol. 104, Dec. 2022, Art. no. 108447.

[30] S. F. Lokman, A. T. Othman, M. H. A. Bakar, and R. Razuwan, "Stacked sparse autoencodersbased outlier discovery for in-vehicle controller area network (CAN)," *Int. J. Eng. Technol*, vol. 7, no. 4.33, pp. 375–380, 2018.

[31] J. Ashraf, A. D. Bakhshi, N. Moustafa, H. Khurshid, A. Javed, and A. Beheshti, "Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4507–4518, Jul. 2021.

[32] G. Kumar and K. Kumar, "AI based supervised classifiers: An analysis for intrusion detection," in *Proc. Int. Conf. Adv. Comput. Artif. Intell.*, Jul. 2011, pp. 170–174.

[33] C. Elkan, "Results of the KDD'99 classifier learning," *ACM SIGKDD Explor. Newslett.*, vol. 1, no. 2, pp. 63–64, Jan. 2000.

[34] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for Internet-of-Things networks," *Future Internet*, vol. 12, no. 3, p. 44, Mar. 2020.

[35] H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Gener. Comput. Syst.*, vol. 122, pp. 130–143, Sep. 2021.

[36] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Comput. Secur.*, vol. 95, Aug. 2020, Art. no. 101851.

[37] H. K. Kim, "Car-hacking dataset," Hacking Countermeasure Res. Lab, South Korea, 2021. [Online]. Available: https://ocslab.hksecurity.net/Datasets/car-hacking-dataset

[38] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.

[39] K. Thakur, H. Alqahtani, and G. Kumar, "An intelligent algorithmically generated domain detection system," *Comput. Electr. Eng.*, vol. 92, Jun. 2021, Art. no. 107129.

[40] G. Kumar, "An improved ensemble approach for effective intrusion detection," *J. Supercomput.*, vol. 76, no. 1, pp. 275–291, Jan. 2020.

[41] G. Kumar, "Evaluation metrics for intrusion detection systems—A study," *Int. J. Comput. Sci. Mobile Appl.*, vol. 2, no. 11, pp. 11–17, Nov. 2014.

[42] H. Alqahtani, M. Kavakli-Thorne, G. Kumar, and F. Sbsstc, "An analysis of evaluation metrics of GANs," in *Proc. Int. Conf. Inf. Technol. Appl. (ICITA)*, 2019.

[43] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *J. Chemometrics*, vol. 18, no. 6, pp. 275–285, Jun. 2004.

[44] M. Pal, "Random forest classifier for remote sensing classification," *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, Jan. 2005.

[45] A. A. Taha and S. J. Malebary, "An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine," *IEEE Access*, vol. 8, pp. 25579–25587, 2020.

[46] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 97, 1997, pp. 211–218.

[47] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

ALI ALTALBE received the M.Sc. degree in information technology from Flinders University, Australia, and the Ph.D. degree in information technology from The University of Queensland, Australia. He is currently an Associate Professor with the Department of IT.

• • •