Complex & Intelligent Systems
https://doi.org/10.1007/s40747-022-00705-w

ORIGINAL ARTICLE

A federated learning framework for cyberattack detection in vehicular sensor networks

Maha Driss[1,2] · Iman Almomani[1,3] · Zil e Huma[4] · Jawad Ahmad[5]

Received: 11 January 2022 / Accepted: 20 February 2022
© The Author(s) 2022

Abstract
Vehicular Sensor Networks (VSN) introduced a new paradigm for modern transportation systems by improving traffic management and comfort. However, the increasing adoption of smart sensing technologies with the Internet of Things (IoT) made VSN a high-value target for cybercriminals. In recent years, Machine Learning (ML) and Deep Learning (DL) techniques attracted the research community to develop security solutions for IoT networks. Traditional ML and DL approaches that operate with data stored on a centralized server raise major privacy problems for user data. On the other hand, the resource-constrained nature of a smart sensing network demands lightweight security solutions. To address these issues, this article proposes a Federated Learning (FL)-based attack detection framework for VSN. The proposed scheme utilizes a group of Gated Recurrent Units (GRU) with a Random Forest (RF)-based ensembler unit. The effectiveness of the suggested framework is investigated through multiple performance metrics. Experimental findings indicate that the proposed FL approach successfully detected the cyberattacks in VSN with the highest accuracy of 99.52%. The other performance scores, precision, recall, and F1 are attained as 99.77%, 99.54%, and 99.65%, respectively.

Keywords Cybersecurity · Internet of things · Intrusion detection · Vehicular sensor networks

Introduction

The rapid advancements in vehicular technologies introduced a number of innovative and intelligent sensors for deployment in modern vehicles [1]. These sensors help the drivers to recognize traffic and road signs efficiently, monitor the roadways, reduce collision risks, and provide an accurate estimation of the distance between the vehicle and surrounding objects [2]. Vehicular Sensor Networks (VSN) offer a smart channel to establish communication between vehicular sensors and Roadside Units (RSUs). The rapid growth of VSN, the complexity of the architecture, the diversity of communication, and the high mobility of vehicles make these networks vulnerable to several cyberattacks [3]. These attacks can vary according to the deployed sensing technologies in VSN. The cyberattacks in VSN can be categorized into inter-vehicle attacks and intra-vehicles attacks. The inter-vehicle attacks mainly target the communication among vehicles and RSUs, and intra-vehicle attacks aim to destroy the communication among smart devices within a vehicle. Inter-vehicle attacks are usually considered to be more dangerous as compared to intra-vehicle attacks [4].

To mitigate these cyberattacks, an Intrusion Detection System (IDS) is one of the most popular and effective security approaches that prevent multiple types of cyberattacks in VSNs [5]. Through the communication of smart sensors with RSUs and other vehicles, IDS provides a powerful

✉ Jawad Ahmad
j.ahmad@napier.ac.uk

Maha Driss
mdriss@psu.edu.sa

Iman Almomani
imomani@psu.edu.sa

Zil e Huma
zhuma.msee18seecs@seecs.edu.pk

1 Security Engineering Lab, CCIS, Prince Sultan University, Riyadh, Saudi Arabia

2 RIADI Laboratory, University of Manouba, Manouba, Tunisia

3 Computer Science Department, King Abdullah II School of Information Technology, The University of Jordan, Amman, Jordan

4 Department of Electrical Engineering, Institute of Space Technology, Islamabad, Pakistan

5 School of Computing, Edinburgh Napier University, Edinburgh, UK

Published online: 24 March 2022

monitoring capacity to recognize suspicious activities and information sharing [6]. In this context, Machine Learning (ML) and Deep Learning (DL) techniques utilized with massive sensor datasets can produce promising results [7,8]. These techniques have been extensively used for the development of IDSs for VSNs [9]. Here, we present some latest studies leveraging ML/DL approaches for attack detection in VSNs. Zhang et al. [10] introduced a privacy-preserving ML-based collaborative IDS for vehicular ad-hoc networks (VANET). The suggested scheme deploys the alternating direction framework of multipliers to a class of empirical risk minimization problems for intrusion detection in a vehicular network. Researchers investigated their model by conducting extensive experiments on the NSL-KDD dataset. Alladi et al. [11] presented an Artificial Intelligence (AI)-based IDS for vehicular networks. The proposed model includes Deep Learning Engines (DLEs) for detecting and classifying cyberattacks in road traffic. The authors in this work deployed these DLEs on multi-access edge computing servers. The experimental findings demonstrated the effectiveness of the proposed framework. Dealing with a massive amount of constantly growing vehicular data is a critical challenge for IDSs. In this context, Bangui et al. [12] developed a Random Forest (RF) algorithm and a posterior detection-based model to improve the attack detection efficiency of IDSs. The obtained results indicated that the proposed model significantly enhanced the cyberattack detection accuracy compared to classical IDSs. Raja et al. [13] proposed a secure and private collaborative IDS to mitigate the security concerns in VANETs. In the proposed scheme, a distributed ML model utilized the potential of intra-vehicle collaboration in the learning process to enhance the scalability, accuracy, and efficiency of the proposed IDS. The performance of the suggested approach was also compared with several ML classifiers. The simulation results showed that the suggested scheme outperformed the existing proposed methods. Ashraf et al. [14] presented a DL-based IDS for Intelligent Transportation Systems (ITS). The authors in this work deployed a Long Short-Term Memory (LSTM) autoencoder algorithm to recognize the malicious activities in vehicular networks. The suggested scheme was evaluated using the UNSW-NB15 dataset. The experimental results proved the higher attack detection efficiency of the proposed scheme compared to eight well-known ML-based IDSs.

Limitations of existing studies

Several improvements have been made to enhance the IDSs for vehicular networks in the studies mentioned above. However, traditional ML-based IDSs face several inherent challenges in modern vehicle security applications, as described below [15]:

- Large volumes of data must be sent from sensors to a distant server, which necessitates additional network traffic encoding and transmission time. Consequently, low bandwidth may result in poor data transmission efficiency. Furthermore, cloud servers are frequently located distant from sensors, requiring data to transit via several edge nodes. As a result of the long-distance data transmission, a VSN with multiple sensor nodes is unable to achieve real-time and high Quality of Service (QoS) expectations. As a result, the typical cloud-based architecture used with the traditional-based IDSs is unsuitable for meeting the aforementioned objectives.
- In the traditional centralized model-training systems, clients communicate their datasets to the cloud server via various communication network links. As a result, wireless communications and core network connections between clients and servers have a significant impact on DL model training and resulting decisions. Consequently, even when the network is down, the connection must be reasonably robust. However, due to the unpredictable wireless connection between the client and server, a centralized design confronts system performance deterioration and probable failures, which can have a substantial impact on the model training and its inferences.
- Since clients must exchange their raw data with other parties, such as cloud or edge servers, to train a model, traditional centralized ML-based IDSs are prone to sensitive data privacy violations and attackers. To address this issue, a tailored set of controls and methodologies identifying the relative relevance of datasets, their sensitivity, compliance requirements, and the application of suitable measures to secure these resources is necessary. These solutions are conceivable, but they necessitate the incorporation of additional resources to the traditional ML-based IDSs, as well as a higher computational cost.
- As data owners become more worried about their privacy, administrative regulations must be implemented to limit data collection to those that are participating in the processing and have been granted explicit approval from the owners. The traditional centralized model-training architecture cannot provide privacy legislation, since clients must submit raw data to the server for model training.

Motivation

The paradigm of federated learning (FL) was proposed by Google researchers in 2016 as a viable alternative for tackling communication costs, data privacy, and regulatory issues [15]. An FL approach is a distributed ML approach in which models are trained on end devices without sharing their local datasets under centralized management. This protects data privacy during the training phase. An edge server or cloud

server collects the learned parameters on a regular basis to construct and update a newer, more accurate model, which is then delivered back to the edge devices for local training. The FL training process is divided into five steps [15]. The FL server first selects an ML model to be trained locally on each client node. Second, at random or using appropriate selection techniques, a subset of current client nodes is picked. Third, the server transmits the initial global model to the client nodes that are selected. Clients download the model's current global parameters and train the model locally. In the fourth phase, each client node transmits changes to the server. Finally, without accessing any clients' data, the FL server gets the changes and aggregates them using aggregation algorithms to produce a new global model. In every round, the FL server orchestrates the training process and sends the global model changes to the selected client nodes. The process is repeated until the desired quality performance is attained.

FL can be an appropriate choice to address the issues encountered with the traditional ML-based IDSs. FL is one of the most adaptable techniques that allow the training of ML algorithms on edge devices [16]. FL approach enables multiple participants to develop robust and efficient ML models without data sharing. Because this strategy preserves the privacy of user data, it is regarded as a better option than non-FL approaches [17]. There are several advantages of FL algorithms. First, FL facilitates the edge devices to learn from predictive models and maintain the training dataset instead of storing it on a centralized server [18]. Second, it saves the data locally on customized service, ensuring data security management. Third, it offers the real-time up-gradation of ML models, because the data are available on edge devices. This feature reduces time consumption, and data can be accessed without contacting the centralized server. Fourth, it is highly suitable for deployments on resource-constrained hardware because of its low complexity and distributed nature [18].

Major contributions

This article proposes a novel federated learning-based architecture for cyberattack detection in the VSN. This framework enables the on-device training of the proposed attack detection model for VSN. The major contributions of this study are summarized in the following points.

1. This article presents an overview of the widely used sensing technologies and potential cyberattacks in VSNs.
2. A Federated learning (FL)-based technique is proposed for cyberattack detection in VSN. In the proposed approach, a group of Gated Recurrent Units (GRUs) with an ensembler unit are deployed to ensure higher attack detection accuracy in VSN.
3. Extensive experiments on a newly generated dataset "Car Hacking: Attack & Defense Challenge 2020 Dataset" are conducted to train the proposed algorithm.
4. The effectiveness of the suggested technique is analyzed using several performance metrics, including accuracy, precision, recall, F1 score, and training time.

The remainder of this article is organized as follows. "Cyberattacks in Vehicular Sensor Networks" investigates the widely used sensing technologies and cyberattacks in VSN. "Proposed Federated Learning Framework" describes the proposed framework along with the mathematical background of the utilized algorithm. "Experiments and Results" comprises implementation platform details, dataset description, and experimental findings discussion. Finally, a brief conclusion with future research directions is presented in "Conclusion".

Cyberattacks in vehicular sensor networks

The VSN contains a number of vehicle sensors that monitor and measure physical parameters related to the vehicle and its environment. These sensors facilitate smooth drive operations and enhance the driver's comfort [19]. Some commonly used sensors of smart vehicles are presented in Fig. 1. All these sensors are made up of advanced electronics and communication technologies. Because of the resource-constrained nature of these sensors, robust and complex security algorithms cannot be directly deployed. Therefore, these sensors are vulnerable to several cyberattacks, as presented in Table 1. The description of some common cyberattacks in VSNs is presented in Table 2. The most prominent vehicular sensing technologies and their security challenges are discussed in the following subsections.

Environmental sensors

These sensors monitor and measure the physical quantities related to vehicular surroundings. The prominent examples of these sensors are camera, Global Positioning System (GPS), ultrasonic sensor, Light image Detection and Ranging

Fig. 1 Commonly used sensors in smart vehicles

Table 1 Commonly used sensors in VSN and relevant cyberattacks

Sensor	Sensor use in vehicles	Cyberattacks
Camera	Detection of traffic signs, improvement in night vision, estimation of collision risks, and parking assistance	Blinding and auto-control
GPS	Navigation and anti-theft purposes	Jamming, spoofing, and blackhole
Ultrasonic sensors	Low-speed maneuvers	Sensor interference, blind spot exploitation, cloaking, physical tampering, and acoustic cancellation
LiDAR	Collision avoidance system and adaptive cruise control	Denial of Service (DoS), replay, spoofing, jamming, and blinding
Radar	Adaptive cruise control and lane change assistant	Jamming and spoofing
Inertial sensors	Information about the vehicle's acceleration and orientation	Spoofing and acoustic cancellation
Magnetic encoders	Measurement of vehicle's speed and angular position	Disruptive attacks and spoofing
TPMS	Monitoring of tire air pressure	Spoofing, eavesdropping, and reverse-engineering

Table 2 Description of cyberattacks in VSNs

Attack	Description
DoS	A DoS attack is a sort of cyberattack in which a malicious actor attempts to disable the regular operation of a device to make it unavailable to its intended users
Jamming	Jamming attacks are a subset of denial-of-service attacks in which hostile nodes disrupt legitimate communication by interfering with networks
Spoofing	Spoofing, in the context of cybersecurity, occurs when someone or something impersonates another entity to win our trust, obtain access to our systems, steal data, steal money, or distribute malware
Acoustic	Acoustic is a type of side-channel attack that exploits sounds emitted by computers or smart devices
Eavesdropping	Eavesdropping is a technique in which an attacker listens passively to network conversations to get private information such as node identifying numbers, routing changes, or application-sensitive data
Blackhole	A blackhole attack occurs when a router deletes all messages that it is supposed to forward. A router can be set improperly to give a zero-cost route to every destination on the Internet on rare occasions
Cloaking	Malicious websites frequently impersonate well-known businesses to house malware and conduct social engineering attacks to obtain user credentials. Certain types of websites frequently seek to conceal hazardous information from search engine crawlers while displaying it to users/client browsers, a practice known as cloaking
Replay	A replay attack occurs when a hacker eavesdrops on a secure network connection, intercepts it, and then fraudulently delays or resends it to trick the receiver into doing what the hacker wants

(LiDAR), and Radio Detection and Ranging (Radar) systems. This section discusses the sensors mentioned above and their vulnerabilities to cyberattacks.

Camera

These are the commonly used sensors in autonomous vehicles to identify their surroundings. These sensors are primarily utilized to identify traffic and road signs, monitor the nearby obstacles, and help to avoid collisions while parking. The major cyberattacks against these sensors include blinding and auto-control attacks [20].

GPS

It is an essential system of autonomous vehicles that facilitates the identification of geographic locations. GPS satellites transmit the navigation signals to the on-ground receivers. Receivers determine the vehicle's current location by computing their distance to at least four different satellites. GPS communication is vulnerable to jamming, spoofing, and blackhole attacks [21].

Ultrasonic sensor

This sensor is used to detect a short-range obstacle and also calculate its distance to the vehicle. This sensor transmits an ultrasonic signal towards the nearby objects. The delay between the transmission and reception of the signal is utilized to compute the precise distance of the vehicle from an obstacle when the signal reflects back from the object. Ultrasonic sensors are generally vulnerable to sensor interference, blind spot exploitation, cloaking, physical tampering, and acoustic cancellation attacks [22].

LiDAR

It generates a 3D map of vehicle surroundings using laser scanning techniques. LiDAR can generate a map of the vehicle's surroundings by transmitting out laser pulses in the scanning process. When these pulses are reflected, LiDAR calculates vehicles' distances to surrounding objects. The prominent cyberattacks on the LiDAR system are Denial of Service (DoS), spoofing, replay, jamming, and blinding attacks [23].

Radar

It transmits electromagnetic signals and measures the distance of nearby objects to vehicles. These sensors determine the distance by calculating the time elapsed from the transmission of a signal to the detection by radar receivers. Most of the radar systems operate within the millimeter-wave frequency band. A short-range radar sensor helps the driver to identify obstacles while parking. Medium-range radars are used in lane change assistance mechanisms, and long-range radars are mostly used in adaptive cruise control. These sensors are usually vulnerable to jamming and spoofing attacks [24].

Vehicle dynamics sensors

Vehicle dynamic sensors provide the measurements of the vehicle's state. These sensors include inertial sensors, magnetic encoders, and tire pressure monitoring systems. In the following subsections, we discuss the sensors mentioned above and their vulnerabilities to cyberattacks.

Inertial sensors

These sensors contain accelerometers and gyroscope sensors. Accelerometer measures the acceleration of the moving objects. Gyroscope sensors calculate the rate of rotation regarding a specific axis. The major cyberattacks on inertial sensors are spoofing and acoustic attacks [25].

Magnetic encoders

The magnetic encoder calculates the angular velocity of vehicle gear. This sensor can measure the wheel's rotational speed using hall effect sensors. Also, it is frequently used with anti-lock braking systems. The magnetic encoder can also be indirectly used with a tire pressure monitoring system to determine the rotational speeds and estimate the difference in pressure values. These sensors are generally vulnerable to disruptive and spoofing attacks [26].

Tire pressure monitoring systems

The tire pressure monitoring system contains four pressure sensors and an Electronics Control Unit (ECU). ECU collects the information from pressure sensors and transmits it to the vehicle's central control unit, along with sensor ID and pressure and temperature measurements. Tire pressure monitoring systems are mainly vulnerable to spoofing, eavesdropping, and reverse-engineering attacks [27].

Proposed federated learning framework

This section presents the adopted ML model and the modules of the proposed framework.

Gated recurrent unit (GRU)

GRU is one of the most popular variants of the recurrent neural network (RNN) [28]. It is also regarded as a simpler version of Long Short-Term Memory (LSTM) because of its lower resources and computational requirements [29]. The proposed cyberattack detection approach is built using the GRU neural network, which takes into consideration the temporal relationship between traffic samples, which is an essential classification feature that allows improving the model's overall detection capability and speed [30]. GRU can also predict time series data and detect unknowable attack patterns [30].

The basic architecture of GRUs contains multiple gates that observe the information flow and regulate the learning process. The gates act as switches that help retain long- and short-term information in the network. Intrusion detection, speech synthesis, speech recognition, and text generation are some real-world deployments of GRUs [31]. The basic architecture of GRU is presented in Fig. 2. The main units of GRUs are discussed in the following.

Sigmoid function

The sigmoid function defines a way to decide which information should be kept or discarded. It generates scores between 0 and 1. If the score is near 0, it enables the network to discard the information. In the case of $\sigma = 1$, it indicates that this information should be retained for future purposes.

Hyperbolic tangent function

This activation function is also referred to simply as the tanh that produces the numbers between -1 and $+1$. The function accepts any real value as input and generates outputs from -1 to 1. The tanh function is often employed in hidden layers,

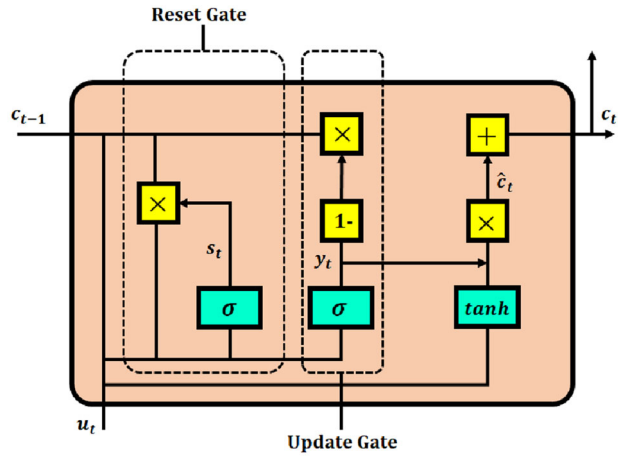

Fig. 2 Basic architecture of GRU

since its average value makes the training process easier for the subsequent layers.

Gates of GRUs

In GRUs, the inputs to each memory cell are concatenated to form a single value, and the architecture works better with only two gates, named reset and update gates. GRUs require less training time and are computationally inexpensive.

(i) Reset gate: If the information is not beneficial for future aspects in terms of (1), then this gate discards the information

$$s_t = \sigma\left(W_s\left[c_{t-1}, u_t\right]\right). \tag{1}$$

Here, s_t determines the sigmoid layer results for the present memory cell, W_s is the weight of s, c_{t-1} represents the information from the previous cell, and u_t indicates the input for the present cell, respectively.

(ii) Update gate: GRUs employ a single gate referred to as an update gate. It decides whether the information from the current state should be kept or discarded using Eqs. (2)–(4)

$$y_t = \sigma\left(W_y\left[c_{t-1}, u_t\right]\right) \tag{2}$$

$$\hat{c}_t = \tanh\left(W\left[s_t * c_{t-1}, u_t\right]\right) \tag{3}$$

$$c_t = \left(1 - y_t * c_{t-1} + y_t * \hat{c}_t\right). \tag{4}$$

Here, y_t is sigmoid layer result, $\hat{c}_t$ is the vector generated by the $tanh$ layer, and c_{t-1} represents previous cell's state.

We use five different input and window sizes for each GRU in the proposed approach. The selection of the appropriate window size is a critical task because of the varying range of data for each window size. The window size contributes to optimizing the ML model performance. The window size increases the training time, because the information retention time increases in each memory cell. The same goes for the selection of hyperparameters. There are no strict rules to determine the optimal relationship between window size and model performance. Same as the hyperparameters selection, there are no hard and fast rules to determine the optimum relationship between window size and model performance.

Cyberattack detection framework

In this work, a novel FL-based scheme is proposed for cyberattack detection in the VSNs. A high-level architecture of the proposed scheme is presented in Fig. 3. The proposed framework contains several modules including virtual IoT prototypes that represent the edge IoT devices and sensors in VSN, a local learning model for each virtual prototype, FL averaging module with a centralized server, a global model for defined window sizes, and an ensembler unit. A detailed description of the aforementioned modules is presented in the following.

Virtual prototypes

A replica model of VSN is built up by creating virtual prototypes. First virtual prototypes fl_n are created for the selected n number of edge devices. In the second stage, some dedicated prototypes fl_{avg} are created that enable the sharing of model parameters of trained ML algorithm among edge devices and the centralized FL server. The used datasets are divided into n blocks, and each one is shared with fl_n.

Preprocessing

Data preprocessing is an important stage for the optimum training of ML/DL models. It makes the captured data best suitable for the input of the neural network. The captured data are first converted to CSV files in the proposed scheme. Then, unnecessary features that do not significantly contribute to the training process are eliminated. Finally, the processed data are split into n blocks and distributed between the virtual prototypes fl_n of edge devices.

FL training

The training procedure is carried out asynchronously. Each client node executes the learning algorithm with its copy of the dataset and shares the weights of the trained local model with fl_{avg} aggregating instance. In this study, 5 GRUs are used with different hyperparameters. The training process of the FL paradigm is detailed in Table 3.

Ensembler

It provides an effective method for combining the outputs of the ML model to achieve a high accuracy score [32,33]. In

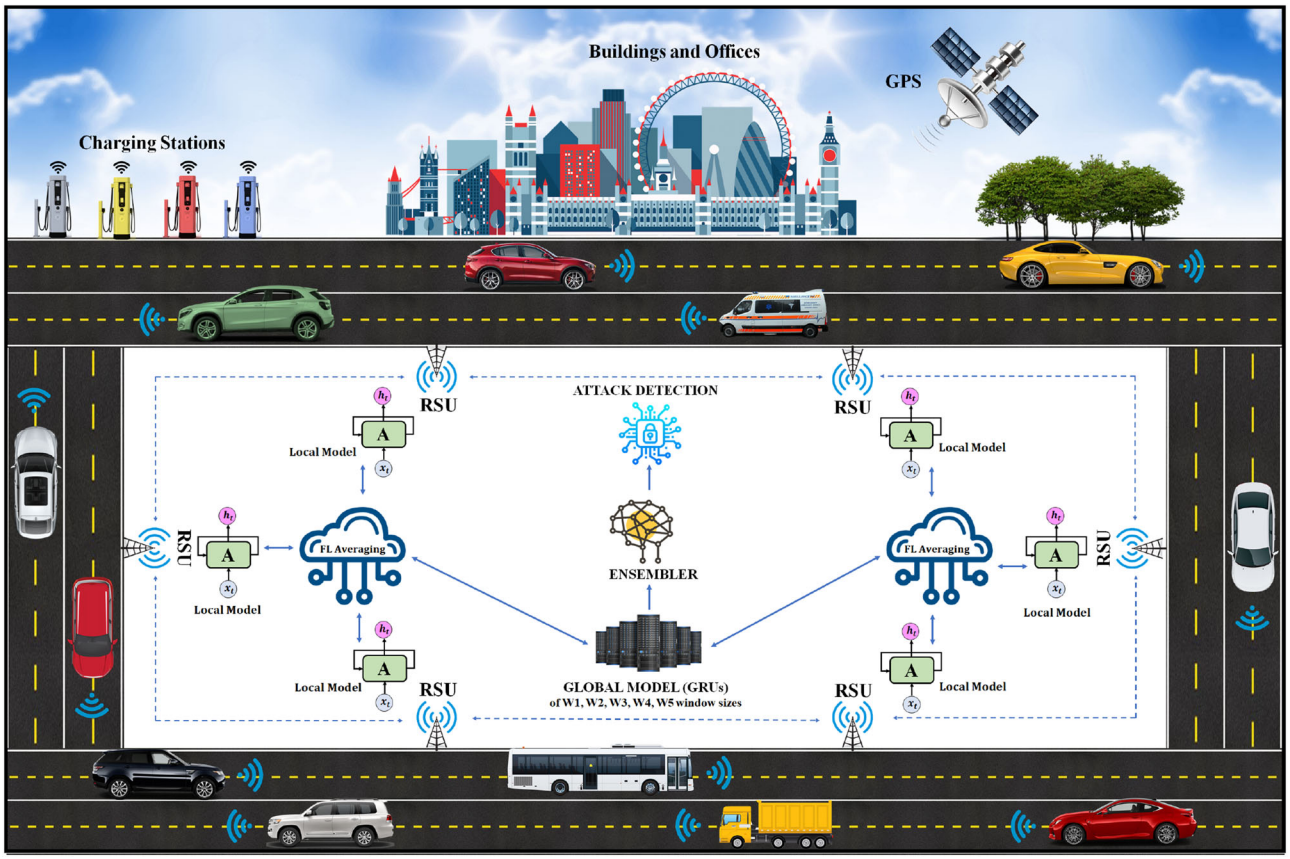

Fig. 3 High-level architecture of the proposed attack detection scheme for VSN

Table 3 Detailed description of the FL training process

FL Training process	
	Begin
Step-1	Initialize the specific window size W_i
Step-2	Define virtual prototypes to represent IoT edge devices fl_i
Step-3	Define the model parameters GRU_{ML} for each window size W_i
Step-4	Share the weights of the trained model with each fl_i
Step-5	At each fl_i, the learning algorithm is executed on GRU_{ML} and after that, the updated weights of the trained model are shared with fl_{avg}
Step-6	fl_i serves as an accumulating unit on the centralized server and monitors the local model updates. The global ML model G_{wi} of each window size is obtained by accumulating the weights of the local model
Step-7	A copy of the global ML model is shared with each edge device
	End

many cases, this is due to the well-established idea of integrating multiple ML models to achieve better performance results than a single ML model. In the proposed framework, we used Random Forest (RF) classifiers to ensemble the global ML models G_{wi}. The RF classifier was chosen due to its numerous advantages, which are listed below.

- It reduces overfitting in decision trees and helps to improve accuracy.
- It can address both classification and regression problems.
- It can handle both categorical and continuous data.
- It automates the replacement of missing values present in the trained data.

– Data normalization is not required, since RF uses a rule-based approach.

For the input data $U = U_1, \ldots, U_n$ with n chunks, each G_{wi} predicts the probabilities values $h_1, h_2, \ldots, h_n$ of each label O for a given input U. The ensembler combines the probabilities of G_{wi} to formulate an ensemble prediction function $p(u)$. The prediction probability h_i can be calculated for the given input data U by using (5)

$$h_i = \tilde{o}_i \left(MG_{wi}(U)\right) \tag{5}$$

$$p(u) = \text{argmax} \sum_{m=1}^{M} J\left(o = h_j(u)\right). \tag{6}$$

According to (6), the $p(u)$ of RF obtains input from probability scores of all ML models G_{wi} for every label. RF treats each probability score as a vote from G_{wi} and predicts the label with a higher confidence score as an output.

Experiments and results

This section presents the details of the simulation platform, dataset description, and evaluation parameters. Additionally, it provides a discussion on the outcomes of the proposed scheme.

Simulation platform

The proposed algorithm is implemented, and its performance is investigated using a desktop computer that has the following characteristics: a 11th Gen Intel®Core™ i9-11900H @ 2.50GHz processor and a 32 GB RAM. An NVIDIA GeForce RTX 3080 Ti 16 GB graphics card is used to facilitate the smooth training process of the proposed FL scheme. The proposed scheme is implemented and simulated in a Python-based environment with the Keras and TensorFlow backend.

Dataset

The proposed framework is analyzed using the latest dataset that was collected and presented by Kang et al. [34] in the "Car Hacking: Attack & Defense Challenge" competition that was organized in 2020. This dataset is an extended version of the previously published "Car Hacking" dataset [35]. The competition's goal was to improve attack and detection techniques for the Controller Area Network (CAN), an extensively utilized standard in-vehicle network. The Hyundai Avante CN7 was the competition's target vehicle. As a result, the dataset consists of Avante CN7 CAN network traffic, which includes both normal and attack messages. The following items are included in the dataset: (1) the initial round train/test dataset and (2) the last round dataset of the host's attack session. This dataset includes 1,270,310 samples, in which 1,090,312 are normal values and 179,998 are anomalous values. This dataset comprises five classes: normal, flooding, spoofing, replay, and fuzzing.

Table 4 The hyperparameters of the GRU models

Parameters	GRU model				
	GRU 1	GRU 2	GRU 3	GRU 4	GRU 5
Learning rate	0.001	0.005	0.01	0.05	0.10
Optimizer	Adam	Adam	Adam	Adam	Adam
Epochs	100	100	100	100	100
Batch size	256	512	512	128	256
Momentum	0.5	0.6	0.7	0.8	0.9
Dropout	0.01	0.05	0.01	0.0	0.05

Hyperparameters

The hyperparameters are the primary parameters that define the neural network's structure and regulate the learning process. In our work, the core architecture of GRUs is fixed. Ranges of appropriate hyperparameters are found by conducting extensive experiments on the "Car Hacking" dataset with a wide range of hyperparameters to assure the best outcomes of the proposed scheme. We selected specific ranges of these hyperparameters through the hit-and-trial method. This method is rigorous and has been widely employed in numerous and various recent research proposing ML-based solutions [36–38], since optimization algorithms and techniques require an additional computational cost to be carried out. Table 4 lists the hyperparameters that were used in each GRU model. The subsections that follow provide a brief summary of the hyperparameters that were used.

Learning rate

This hyperparameter controls the training speed of the ML model. The selection of an accurate learning rate is a critical task. A low learning rate can efficiently train a model, but learning speed will be slow, and the model can also get stuck [39]. On the other hand, a high learning rate speeds up training, but can lead to multiple output errors. In our experiments, we defined five learning rate values: 0.001, 0.005, 0.01, 0.05, and 0.10.

Optimizer

It is an algorithm used to minimize the loss function or maximize production efficiency. Optimizers are mathematical functions that depend on model parameters such as weights and biases. These algorithms help determine the

change in weights to minimize errors. We used "Adam" in our experiments, one of the most used optimizers. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. There are various appealing advantages of adopting Adam, which are as follows [40]:

- It is straightforward to implement.
- It is computationally efficient.
- Its memory requirements are minimal.
- Its gradients are invariant to diagonal rescaling.
- It is perfectly adapted to solve problems handling a significant amount of data and/or parameters.
- It is appropriate for problems involving highly noisy/or sparse gradients.
- The used hyperparameters have intuitive interpretation and require minimal fine-tuning.

Epochs

This parameter defines one complete execution of the ML algorithm. The appropriate selection of the number of epochs is a critical task. With the completion of each epoch, the model parameters of the ML model are updated. In our experiments, we used 100 epochs for each GRU model.

Batch size

This hyperparameter presents the total number of samples present in a single mini-batch. A very small batch size selection can cause a high degree of variance. On the other hand, if the batch size is too large, it may cause overfitting effects. In our experiments, we defined three batch sizes: 128, 256, and 512.

Momentum

This hyperparameter establishes the direction of the next step based on the knowledge of the previous step. It contributes to the resistance of the ML model to oscillations. In our experiments, we set a momentum range from 0.5 to 0.9.

Dropout

It is a regularization technique that approximates the number of neurons from a neural network during the training phase. Dropout enables the model to reduce the overfitting effects, which can help make accurate predictions. In our experiments, the considered dropout values are 0.0, 0.01, and 0.05.

Performance evaluation parameters

The effectiveness of the proposed model was investigated through several evaluation metrics. First, the predicted outputs of trained algorithms were compared with real values. Based on the comparison, True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) were computed. TP and TN indicate the number of correct predictions of the trained model for both attacks and normal behaviours, respectively. FP and FN represent the incorrect predictions of the trained model for both attacks and normal behaviours, respectively. The parameters mentioned above are further utilized to calculate the accuracy, precision, recall, and F1 score.

Accuracy

This metric indicates the percentage of accurate attacks and normal events predictions. It can be easily calculated by dividing the number of accurate predictions by the number of total predictions

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

Precision

It expresses the proportion of accurately anticipated anomalous observations compared to the total number of observations classified as anomalous

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{8}$$

Recall

This metric quantifies the proportion of accurately predicted abnormal values relative to accurately predicted abnormal observations and erroneously predicted normal observations

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{9}$$

F1 score

It is defined as the harmonic mean of the model's precision and recall

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}. \tag{10}$$

Results and discussion

To evaluate the performance of the proposed algorithm, extensive experiments were conducted on "Car Hacking:

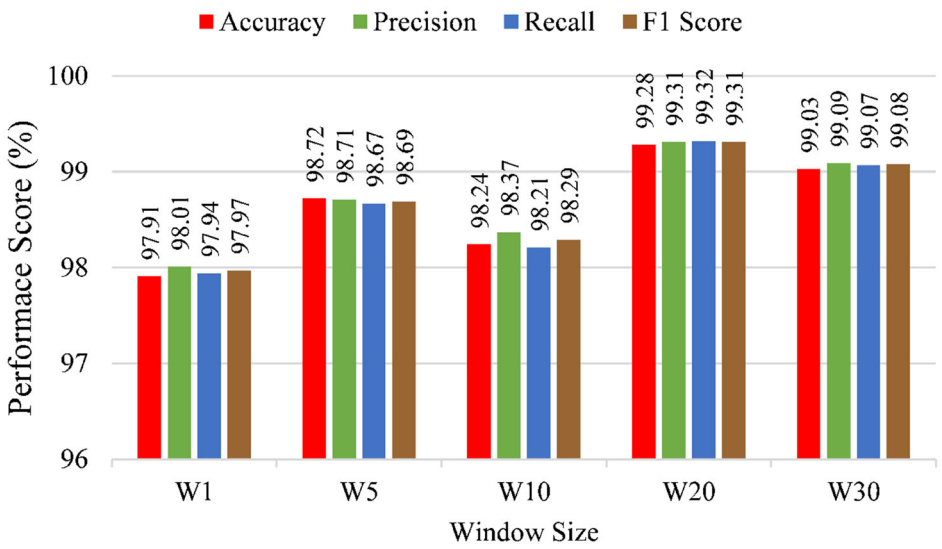

Fig. 4 Performance results of GRU-1 for different window sizes

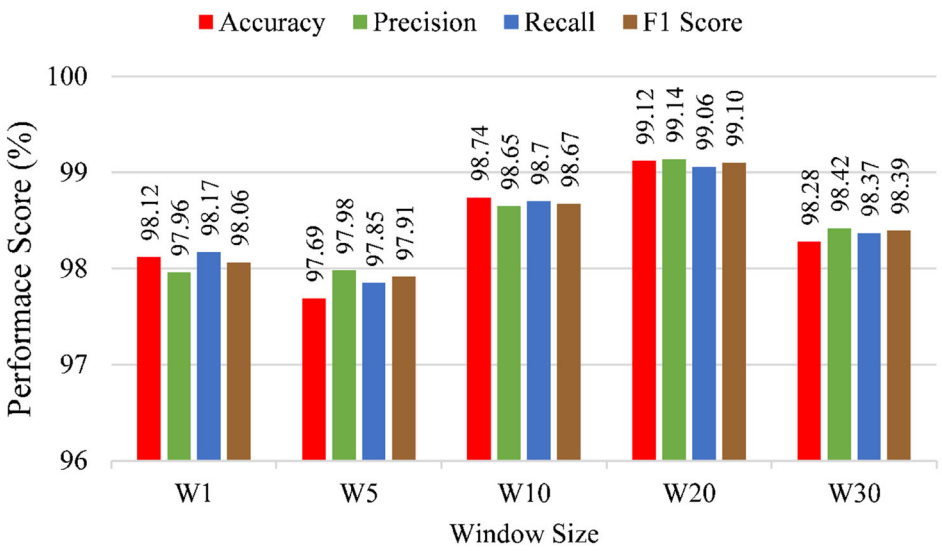

Fig. 5 Performance results of GRU-2 for different window sizes

Attack & Defense Challenge 2020 Dataset". The dataset was split into training and testing datasets with a ratio of 75% and 25%, respectively. As mentioned previously, 5 GRU models were used in the proposed scheme. The hyperparameters of these models are presented in Table 3. The performance of GRU models is evaluated according to different window sizes that are 1, 5, 10, 20, and 30. GRU-1 demonstrated the highest performance at W20. This model achieved the highest detection accuracy of 99.28% for this window size. The other performance scores are also greater than 99% for this window size. For W1, W5, and W10, the performance of GRU-1 is between 97% and 99%. Detailed results of GRU-1 performance for different window sizes are shown in Fig.4.

The performance results of GRU-2 for different window sizes are presented in Fig. 5. GRU-2 also achieved the highest performance at W20. This model achieved the highest detection accuracy of 99.12% for this window size. All the other performance scores are also greater than 99% for this window size. For W1, W5, W10, and W30, the performance results of GRU-2 are between 97% and 98%. The GRU-3 has demonstrated the highest attack detection performance compared to any other GRU. This model achieved the highest detection accuracy of 99.52% for W20. The other performance scores are also greater than 99.50% for this window size. For W1, W5, W10, and W30, the performance results of GRU-3 are between 98% and 99.25%. Performance results of GRU-3 for different window sizes are presented in Fig. 6.

The performance results of the GRU-4 and GRU-5 are lower than those of all other GRUs. These models achieved the highest attack detection accuracies of 98.23% and 98.02% for W20, respectively. For the other window sizes, the performance of both models is between 96% and 97%. The performance results of GRU-4 and GRU-5 for different window sizes are presented in Figs. 7 and 8, respectively.

The results of the simulation proved the satisfactory performance of the GRU models. The average attack detection

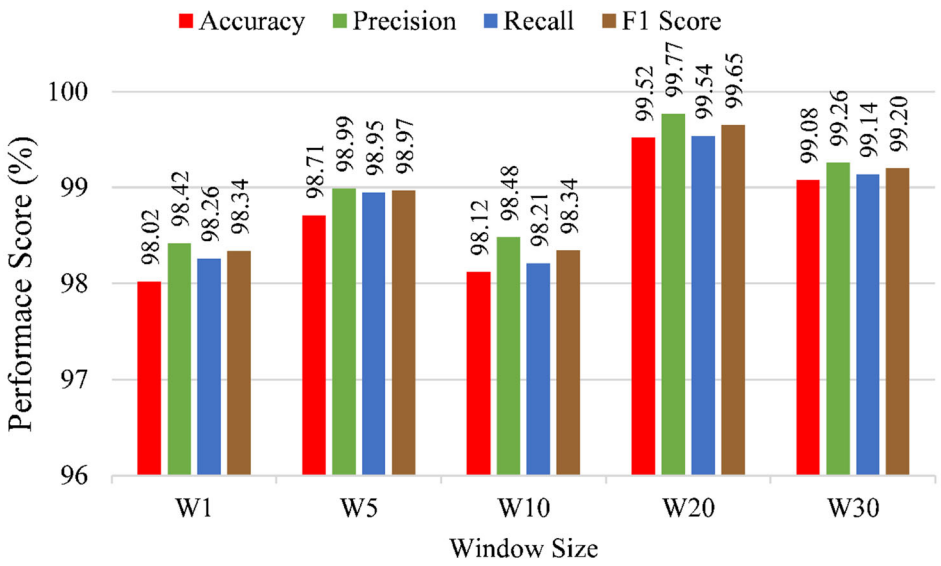

Fig. 6 Performance results of GRU-3 for different window sizes

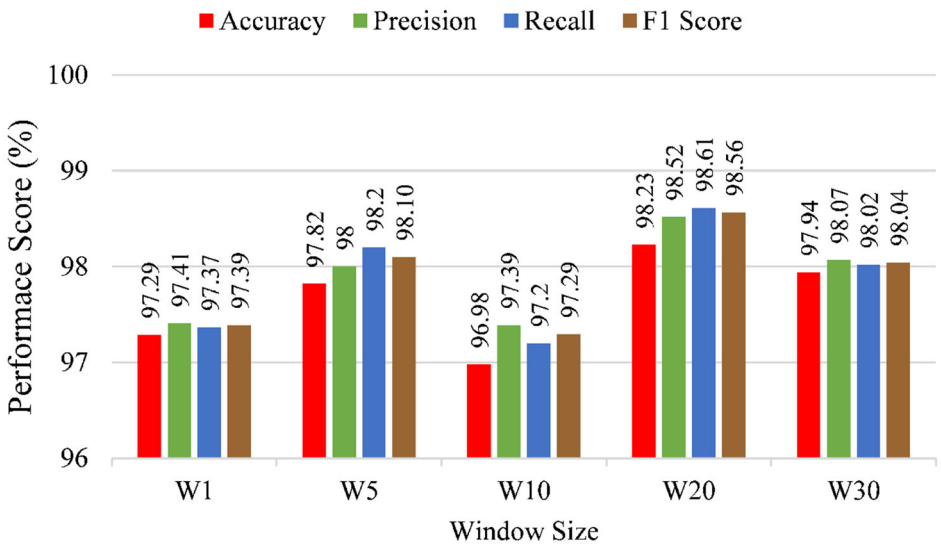

Fig. 7 Performance results of GRU-4 for different window sizes

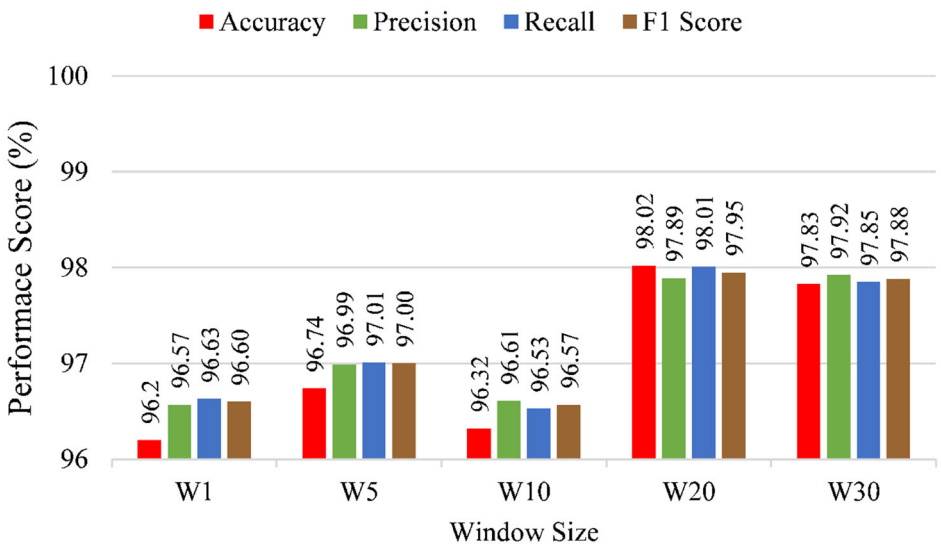

Fig. 8 Performance results of GRU-5 for different window sizes

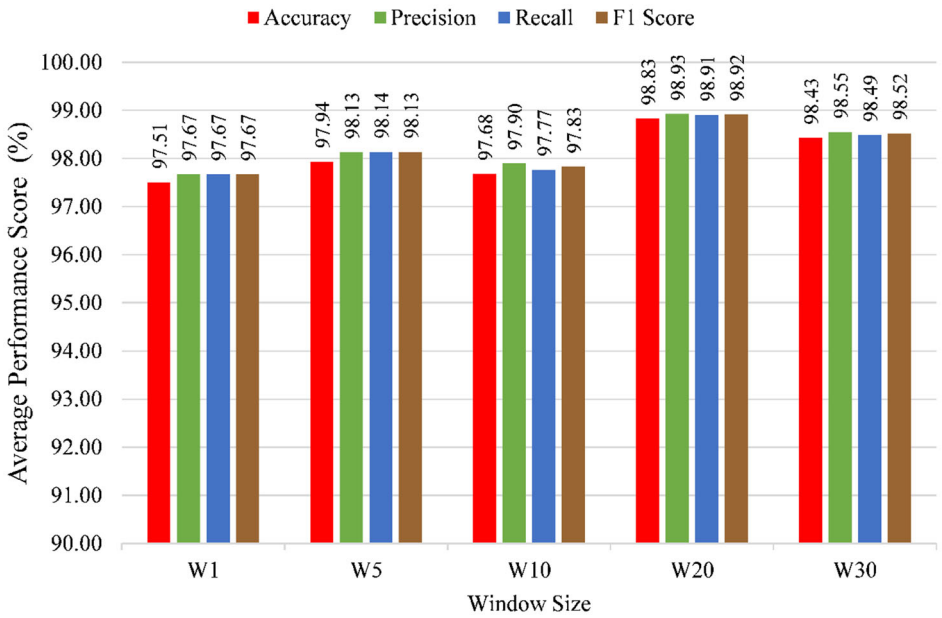

Fig. 9 Average performance of the proposed FL architecture

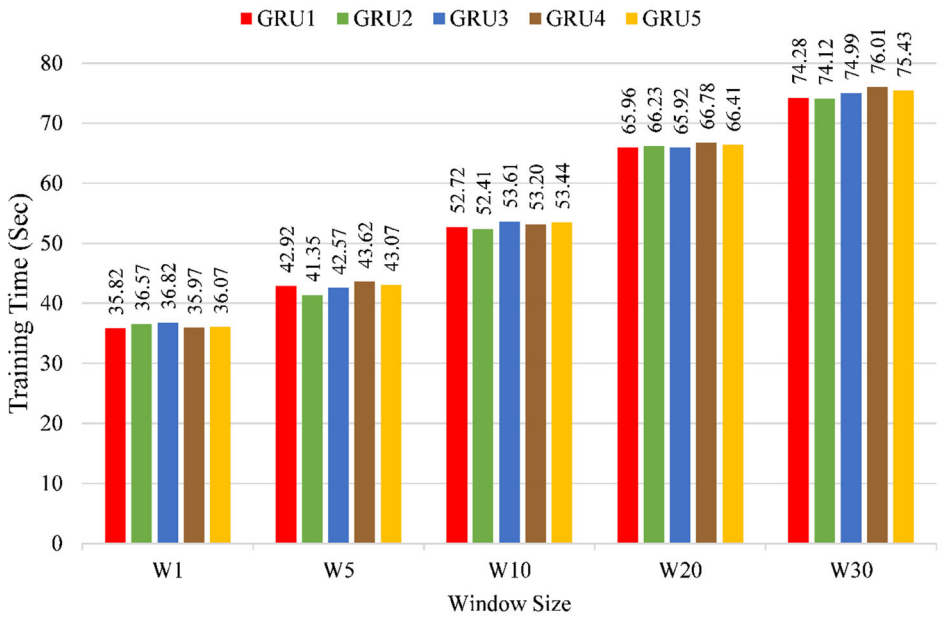

Fig. 10 Training time comparison for different window sizes

Table 5 Performance comparison of the proposed scheme with related studies

Work	Publication year	Proposed scheme	Accuracy
Seo et al. [35]	2018	GAN	98.00%
Ashraf et al. [14]	2020	LSTM autoencoder algorithm	99.00%
Zadid et al. [41]	2020	LSTM	87.90%
Khan et al. [42]	2021	LSTM	99.11%
Abdel-Basset et al. [43]	2021	Two anomaly detection DL models	97.82%
Song et al [44]	2021	LSTM	95.37%
Our work	2022	Federated learning scheme based on GRU and RF ensembler	99.52%

performance of the proposed FL scheme is presented in Fig.9. The average attack detection accuracy achieved by the proposed framework is 98.83%. The other average of precision, recall, and F1 scores reached 98.93%, 98.91%, and 98.92%, respectively. All simulations were run for 100 epochs. The training time of the proposed algorithms was increased with the window sizes. A comparison of the training time for different window sizes is presented in Fig. 10.

To validate the performance of the proposed scheme, we compare the results with recently published related works. The comparison is made against the papers that worked in the same context and used the same dataset, the "Car Hacking" dataset. As it is demonstrated by the performance results provided in Table 5, it is clear that our model achieves the highest accuracy with the extended version of the same dataset among the considered studies that rely on classical IDS-based approaches or centralized DL models. This is justified by the fact that the models are trained independently. In fact, the training is done collaboratively and independently on individual participants by opting for the FL approach. Local epochs, in particular, are defined in the learning parameters, and each participant trains the data by running the local epochs. The local update is computed after a certain number of epochs, and the participants communicate the updates to the cloud server. The cloud server gets each participant's update, averages them, and then aggregates the next global model. The participants carry out the training procedure for the next communication round based on this global model. The process is repeated until the necessary convergence is reached or the communication rounds is completed. This learning process proposed by the FL approach has proven its effectiveness and efficiency in several case studies, in particular, in our case study dealing with the detection of cyberattacks in the VSNs, by reducing the training time and increasing the data accuracy.

Conclusion

This article proposes a federated learning-based framework for efficient cyberattack detection in VSNs. The proposed FL scheme enables the sharing of computational capabilities with on-device training. A group of GRU models with an ensembler unit is used to ensure high attack detection performance. Extensive experiments were conducted on the "Car Hacking: Attack and Defense Challenge 2020" dataset. The performance of the proposed model was analyzed through multiple performance metrics, including accuracy, precision, recall, F1-score, and training time. The experimental findings illustrated that the proposed FL scheme provides accurate and efficient privacy-preserving attack detection in VSNs.

As future work, we plan to undertake additional in-depth experiments in this field, using fusion- and voting-based techniques to deliver more precise attack detection and classification outcomes. Furthermore, we intend to look into the promising field of explainable FL [45], which refers to strategies for providing human-readable insights into data, variables, and decisions. Finally, future developments will use DL models as microservices-based architecture [46–48] to construct our suggested solution in a distributed architectural style. DL as a microservice is a self-contained, locally executable, and easily testable and maintainable software unit that can be reused and adjusted for a wide range of features and datasets by simply altering configuration parameters.

Acknowledgements The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

Declarations

Funding Prince Sultan University.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Ghaleb FA, Maarof MA, Zainal A, Al-rimy BAS, Alsaeedi A, Boulila W (2019) Ensemble-based hybrid context-aware misbehavior detection model for vehicular ad hoc network. Remote Sens 11(23):2852
2. Guo G, Wen S (2015) Communication scheduling and control of a platoon of vehicles in vanets. IEEE Trans Intell Transp Syst 17(6):1551–1563
3. El-Rewini Z, Sadatsharan K, Sugunaraj N, Selvaraj DF, Plathottam SJ, Ranganathan P (2020) Cybersecurity attacks in vehicular sensors. IEEE Sens J 20(22):13752–13767
4. Shu J, Zhou L, Zhang W, Du X, Guizani M (2020) Collaborative intrusion detection for vanets: a deep learning-based distributed sdn approach. IEEE Trans Intell Transp Syst
5. Liang J, Lin Q, Chen J, Zhu Y (2019) A filter model based on hidden generalized mixture transition distribution model for intrusion detection system in vehicle ad hoc networks. IEEE Trans Intell Transp Syst 21(7):2707–2722

6. Sharma S, Kaul A (2018) A survey on intrusion detection systems and honeypot based proactive security mechanisms in vanets and vanet cloud. Vehicular Commun 12:138–164
7. Latif S, Huma Z, Jamal SS, Ahmed F, Ahmad J, Zahid A, Dashtipour K, Aftab MU, Ahmad M, Abbasi QH (2021) Intrusion detection framework for the internet of things using a dense random neural network. IEEE Trans Ind Inf
8. Khan MA, Khan KMA, Latif S, Shah AA, Rehman Mujeeb U, Boulila W, Driss M, Ahmad J (2020) Voting classifier-based intrusion detection for iot networks. In: Advances on Smart and Soft Computing, pp 313–328, Springer
9. Shafique A, Ahmed J, Boulila W, Ghandorh H, Ahmad J, Rehman MU (2020) Detecting the security level of various cryptosystems using machine learning models. IEEE Access 9:9383–9393
10. Zhang T, Zhu Q (2018) Distributed privacy-preserving collaborative intrusion detection systems for vanets. IEEE Trans Signal Inf Process Over Netw 4(1):148–161
11. Alladi T, Kohli V, Chamola V, Yu FR, Guizani M (2021) Artificial intelligence (ai)-empowered intrusion detection architecture for the internet of vehicles. IEEE Wirel Commun 28(3):144–149
12. Bangui H, Ge M, Buhnova B (2021) A hybrid machine learning model for intrusion detection in vanet. Computing, pp 1–29
13. Raja G, Anbalagan S, Vijayaraghavan G, Theerthagiri S, Suryanarayan SV, Wu X-W (2020) Sp-cids: secure and private collaborative ids for vanets. IEEE Trans Intell Transp Syst
14. Ashraf J, Bakhshi AD, Moustafa N, Khurshid H, Javed A, Beheshti A (2020) Novel deep learning-enabled lstm autoencoder architecture for discovering anomalous events from intelligent transportation systems. IEEE Trans Intell Transp Syst
15. Abreha HG, Hayajneh M, Serhani MA (2022) Federated learning in edge computing: a systematic survey. Sensors 22(2):450
16. Hu K, Li Y, Xia M, Wu J, Lu M, Zhang S, Weng L (2021) Federated learning: a distributed shared machine learning method. Complexity
17. Li L, Fan Y, Tse M, Lin K-Y (2020) A review of applications in federated learning. Comput Ind Eng, pp 106854
18. Mothukuri V, Khare P, Reza MP, Seyedamin P, Ali D, Gautam S (2021) Federated learning-based anomaly detection for iot security attacks. IEEE Int Things J
19. Fleming WJ (2008) New automotive sensors-a review. IEEE Sens J 8(11):1900–1921
20. Yan C, Wenyuan X, Liu J (2016) Can you trust autonomous vehicles: contactless attacks against sensors of self-driving vehicle. Def Con 24(8):109
21. Parkinson S, Ward P, Wilson K, Miller J (2017) Cyber threats facing autonomous and connected vehicles: future challenges. IEEE Trans Intell Transp Syst 18(11):2898–2915
22. Wenyuan X, Yan C, Jia W, Ji X, Liu J (2018) Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles. IEEE Int Things J 5(6):5015–5029
23. Petit J, Stottelaar B, Feiri M, Kargl F (2015) Remote attacks on automated vehicles sensors: experiments on camera and lidar. Black Hat Eur 11(2015):995
24. Takefuji Y (2018) Connected vehicle security vulnerabilities [commentary]. IEEE Technol Soc Mag 37(1):15–18
25. Levenberg E (2014) Estimating vehicle speed with embedded inertial sensors. Transp Res Part C: Emerg Technol 46:300–308
26. Shoukry Y, Martin P, Yona Y, Diggavi S, Srivastava M (2015) Pycra: physical challenge-response authentication for active sensors under spoofing attacks. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp 1004–1015
27. Rouf I, Miller RD, Mustafa HA, Taylor T, Oh S, Xu W, Gruteser M, Trappe W, Seskar I (2010) Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In USENIX security symposium, vol 10
28. Rehman S, Khaliq M, Imtiaz SI, Rasool A, Shafiq M, Javed AR, Jalil Z, Ali KB (2021) Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru). Fut Gen Comput Syst 118:453–466
29. Boulila W, Ghandorh H, Khan MA, Ahmed F, Ahmad J(2021) A novel cnn-lstm-based approach to predict urban expansion. Ecol Inf pp 101325
30. Al-Imran Md, Ripon SH (2021) Network intrusion detection: an analytical assessment using deep learning and state-of-the-art machine learning models. Int J Comput Intell Syst 14(1):1–20
31. Li B, Yuhao W, Song J, Rongxing L, Li T, Zhao L (2020) Deepfed: federated deep learning for intrusion detection in industrial cyber-physical systems. IEEE Trans Ind Inf 17(8):5615–5624
32. Zhou Z-H (2021) Ensemble learning. In: Machine learning, Springer, New York, pp 181–210
33. Abbas A, Khan MA, Latif S, Ajaz M, Shah AA, Ahmad J (2021) A new ensemble-based intrusion detection system for internet of things. Arab J Sci Eng pp 1–15
34. Kang H, Kwak BI, Lee YH, Lee H, Lee H, Kim HK (2021) Car hacking and defense competition on in-vehicle network. In: Workshop on automotive and autonomous vehicle security (AutoSec) vol 2021, pp 25
35. Seo E, Song HM, Kim HK (2018) Gids: gan based intrusion detection system for in-vehicle network. pp 1–6
36. Huma ZE, Latif S, Ahmad J, Idrees Z, Ibrar A, Zou Z, Alqahtani F, Baothman F (2021) A hybrid deep random neural network for cyberattack detection in the industrial internet of things. IEEE Access 9:55595–55605
37. Verma S, Misra JP, Singh J, Batra U, Kumar Y (2021) Prediction of tensile behavior of fs welded aa7039 using machine learning. Mater Today Commun 26:101933
38. AlGhamdi R, Aziz A, Alshehri M, Pardasani KR, Aziz T (2021) Deep learning model with ensemble techniques to compute the secondary structure of proteins. J Supercomput 77(5):5104–5119
39. Latif S, Zou Z, Idrees Z, Ahmad J (2020) A novel attack detection scheme for the industrial internet of things using a lightweight random neural network. IEEE Access 8:89337–89350
40. Boulila W, Driss M, Alshanqiti E, Al-Sarem M, Saeed F, Krichen M (2022) Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives. Adv Smart Soft Comput, pp 477–484
41. Abdel-Basset M, Moustafa N, Hawash H, Razzak I, Sallam KM, Elkomy OM (2021) Federated intrusion detection in blockchain-based smart transportation systems. IEEE Trans Intell Transp Syst
42. Khan IA, Moustafa N, Pi D, Haider W, Li B, Jolfaei A (2021) An enhanced multi-stage deep learning framework for detecting malicious activities from autonomous vehicles. IEEE Trans Intell Transp Syst
43. Song HM, Kim HK (2021) Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data. IEEE Trans Veh Technol 70(2):1098–1108
44. Khan Z, Chowdhury M, Islam M, Huang C-Y, Rahman M (2020) Long short-term memory neural network-based attack detection model for in-vehicle network security. IEEE Sens Lett 4(6):1–4
45. Roscher R, Bohn B, Duarte MF, Garcke J (2020) Explainable machine learning for scientific insights and discoveries. IEEE Access 8:42200–42216
46. Driss M (2021) Ws-advising: a reusable and reconfigurable microservices-based platform for effective academic advising. J Ambient Intell Hum Comput pp 1–12
47. Driss M, Hasan D, Boulila W, Ahmad J (2021) Microservices in iot security: current solutions, research challenges, and future directions. Procedia Comput Sci 192:2385–2395. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021

48. Driss M, Atitallah SB, Albalawi A, Boulila W (2021) Reqwscomposer: a novel platform for requirements-driven composition of semantic web services. J Ambient Intell Hum Comput pp 1–17

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.