Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.047079

ARTICLE

# Lightweight Intrusion Detection Using Reservoir Computing

**Jiarui Deng[1,2], Wuqiang Shen[1,3], Yihua Feng[4], Guosheng Lu[5], Guiquan Shen[1,3], Lei Cui[1,3] and Shanxiang Lyu[1,2,*]**

[1]Joint Laboratory on Cyberspace Security, China Southern Power Grid, Guangzhou, 510080, China

[2]College of Cyber Security, Jinan University, Guangzhou, 510632, China

[3]Guangdong Power Grid Company Limited, Guangzhou, 510663, China

[4]School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, 510006, China

[5]China Southern Power Grid, Extra High Voltage Transmission Company, Guangzhou, China

*Corresponding Author: Shanxiang Lyu. Email: lsx07@jnu.edu.cn

Received: 24 October 2023 Accepted: 30 November 2023 Published: 30 January 2024

**ABSTRACT**

The blockchain-empowered Internet of Vehicles (IoV) enables various services and achieves data security and privacy, significantly advancing modern vehicle systems. However, the increased frequency of data transmission and complex network connections among nodes also make them more susceptible to adversarial attacks. As a result, an efficient intrusion detection system (IDS) becomes crucial for securing the IoV environment. Existing IDSs based on convolutional neural networks (CNN) often suffer from high training time and storage requirements. In this paper, we propose a lightweight IDS solution to protect IoV against both intra-vehicle and external threats. Our approach achieves superior performance, as demonstrated by key metrics such as accuracy and precision. Specifically, our method achieves accuracy rates ranging from 99.08% to 100% on the Car-Hacking dataset, with a remarkably short training time.

**KEYWORDS**

Echo state network; intrusion detection system; Internet of Vehicles; reservoir computing

## 1 Introduction

Advances in the Internet of Things (IoT) and vehicular ad-hoc networks (VANETs) have catalyzed a new era of the blockchain-empowered Internet of Vehicles (IoV), which integrates humans, vehicles, things, and environments into a comprehensive network system to provide real-time or long-term services [1–3]. By utilizing established network technologies such as deep learning, and blockchain, IoV can obtain, organize, and process vast amounts of data from both vehicles and external entities, and preserve data security and privacy [4–8], thereby enhancing the computability, extensibility, and sustainability of intricate network systems and information services. In practical terms, IoV can alleviate traffic congestion, prevent accidents, and even address safety concerns caused by the escalating vehicular density.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As an open and integrated network system, the development of IoV relies on the integration of multiple technologies, services, and standards. However, the resulting heterogeneity, coupled with a large number of vehicles, renders IoV vulnerable to cyber-attacks [9,10], highlighting the need for robust IoV security measures. Moreover, due to the inherent characteristics of blockchain-empowered IoT, such as transparency and non-comparability, IoV faces new challenges in terms of security. Attackers can exploit vulnerable connection points to access privacy, manipulate vehicles, and carry out malicious acts, including controlling the brakes, tracking the car, or even shutting it down, resulting in disastrous consequences [9]. Cyber-attacks in IoV can infiltrate two areas: intra-vehicle networks (IVNs) and external vehicular networks. In IVNs, electronic control units (ECUs) exchange information via the controller area network (CAN) bus to execute functions and instructions. Security threats mainly arise from message injection attacks due to the lack of message authentication and the broadcast transmission strategy. Regarding external vehicular networks, there are diverse external entities to interact with, such as pedestrians, smart devices, and infrastructures. The likelihood of vehicles being exposed to outer attacks, which tend to be various and frequent, is significantly amplified.

Countermeasures against cyber-attacks are crucial for the long-term robustness and sustainability of IoV. Intrusion detection systems (IDSs), as essential protection mechanisms, detect potential threats and malicious content both internally and externally by monitoring and analyzing network data such as network traffic, connections, objects, etc., and take security measures immediately. With the advances in machine learning (ML) and deep learning (DL), particularly the success of convolutional neural networks (CNN) in pattern recognition, researchers have explored ML-based IDS and achieved remarkable results in intrusion detection [11–15]. However, applying ML-based IDS to IoV remains challenging due to several reasons. Firstly, deploying IDS in the IoV scenario is subject to hardware constraints such as computing power, memory size, and communication capability. Conventional ML techniques generally consume a large amount of computational resources to train and deploy a model, while the well-trained parameters also occupy substantial storage space. Secondly, IDS, as a safety-critical system, should fulfill real-time and high-precision requirements simultaneously. Some ML-based IDSs overemphasize detection accuracy, resulting in complex and time-consuming methods. Lastly, data preprocessing and feature selection are essential aspects of IDS. IoV involves a high volume of network traffic data, but only a negligible proportion of the total data constitutes malicious content. Therefore, data preprocessing measures should be employed to mitigate the class imbalance problem. Additionally, traffic data contains a plethora of features, and it is crucial to remove irrelevant features to improve detection accuracy and alleviate the computational burden.

This paper proposes an IDS model based on reservoir computing (RC) to protect the IoV system. RC, as a framework for computation, has been successfully applied to address real-world problems in various domains, including time series forecasting, handwriting recognition, and network anomaly detection [16–18]. In contrast to typical recurrent neural networks (RNNs), RC offers computational efficiency with less memory demand, attributed to its architecture and training algorithm. RC can be considered a simplified RNN, with a recurrent (reservoir) topology randomly determined, and only the readout layer requires training using simple linear regression. RC can be employed in online intrusion detection, which deals with streams of large input data in real time by classifying regular and anomalous data points.

The main contributions of this paper are summarized as follows:

- We propose a lightweight IDS for IoV. Unlike existing CNN-based methods, the proposed RC-based method requires significantly less training time and storage space. This is the first time

that RC has been applied to intrusion detection in IoV, addressing the computational resource and memory space limitations of traditional machine learning models and opening up a new avenue for intrusion detection in IoV.
- The proposed scheme converts processed traffic data into images and feeds them into RC for intrusion detection. Additional resampling and feature engineering steps are employed to improve data quality and increase detection accuracy, addressing class imbalance and feature redundancy in external data.
- We validate our scheme using the Car-Hacking dataset [19] and the CICIDS2017 dataset [20], which simulate internal and external IoV networks, respectively. Experimental results demonstrate that our scheme achieves high accuracy requirements with computational efficiency.

The remainder of this paper is organized as follows. Section 2 introduces the basis of RC. Section 3 elaborates on the proposed scheme. Section 4 presents simulation results and discussions. Section 5 concludes this paper.

## 2 Related Work

IDSs for IoV networks are generally classified into two groups, namely IDSs based on traditional machine learning and IDSs based on neural networks. In this section, we provide a brief overview of recent advances in IDSs from these two perspectives.

### *2.1 Traditional ML Algorithms*

The field of IoV has seen numerous research into intrusion detection systems based on traditional machine learning techniques. In a study by [21], various ML methods, including logistic regression, naïve Bayes, decision tree, support vector machine, k-nearest neighbor, random forest, and XGBoost, were validated on a large-scale, heterogeneous dataset known as ToN-IoT. Chi-square and SMOTE were used for data preprocessing. This study concluded that XGBoost outperformed other ML-based methods. The study in [22] focused on DDoS attacks, devising an IDS consisting of a real-time network traffic collection module and a network detection module and evaluating NSL-KDD and UNSW-NB15 datasets. In the context of big data, this work introduced a distributed architecture using Spark and Hadoop Distributed Files Systems to speed up data collection and processing. Random forest was adopted for classification in the detection module. Researchers in [23] built an ensemble learning model for intrusion detection utilizing tree-based ML algorithms, including decision trees, random forests, extra trees, and extreme gradient boosting. To tackle class imbalance and computational costs, SMOTE oversampling and tree-based averaging feature selection techniques were also adopted. Results on CAN intrusion and the CICIDS2017 dataset demonstrated the efficiency of the proposed IDS. In [24], the authors combined Logarithmic Ratio (OBLR), outlier detection, and metric learning to combat dataset imbalance and achieve efficiency. The detection process is implemented by LightGBM after genetic algorithm feature selection. Evaluations were performed on UNSW-NB15 as an external network dataset and ROAD, Car-Hacking CAN-intrusion as intra-vehicle datasets.

### *2.2 Neural Networks*

Compared to traditional ML-based IDSs, deep neural network methods can exploit the nonlinear relationship between features, which cannot be extracted by expert domain and feature selection methods. Reference [13] used a feed-forward neural network for IDS based on multi-layer perceptron, which obtained 99% accuracy on the CICIDS2017 dataset. Convolutional Neural Network (CNN) is an important research direction in DL-based IDS construction. The authors of [11] designed a

1D CNN to protect the in-vehicle CAN bus system. The ID CNN-based classifier achieved 99.99% accuracy on the dataset generated from three car models. By removing the redundant part of the Inception-ResNet architecture, Song et al. [12] proposed a deep CNN (DCNN) model to learn temporal sequential patterns of input data and detect in-vehicle attacks. The proposed DCNN was proved to be effective by experiments compared to other ML-based algorithms. In [25], the authors also trained a 1D CNN model for IDS with the ToN-IoT dataset. The highlight of this work is the application of the SHAP method to explain the performance of the CNN-based IDS. Reference [26] proposed an intelligent IDS (IIDS) for IoV based on a modified CNN model with hyperparameter optimization. Experimental results showed that the proposed IIDS achieved 98% accuracy in detecting attacks. Reference [14] transformed vehicle network traffic data into images for CNN to distinguish attack patterns and achieved satisfying results on Car-Hacking and CICIDS2017 datasets. Among various pre-trained CNN models, the authors preserved the bottom layer and only fine-tuned the top layers based on transfer learning to save training time. From the point of exploiting temporal relationships within network data, Recurrent Neural Network (RNN) and its variation Long Short-Term Memory (LSTM) network have attracted attention. Reference [27] conceptualized a privacy-preserving-based framework for IoV, integrating blockchain and LSMT techniques. They separately ensure secure transmission and intrusion detection. The authors of [28] used a Long Short-Term Memory-AutoEncoder (LSTM-AE) to encode data into a new format for private feature extraction. Attention-based Recurrent Neural Network (A-RNN) was adopted for intrusion classification in the IDS part. The performance of this scheme was validated through ToN-IoT and CICIDS2017 datasets. Reservoir computing (RC), as an emerging and promising paradigm in the realm of RNNs, inherits the ability to process temporal data while getting rid of gradient descent-based training methods, thereby alleviating the computational burden and accelerating convergence speed. Reference [17] utilized echo state network, a type of RC, to discriminate attacks. Experimental results showed that ESN can achieve comparable performance to bidirectional LSTM with shorter training time. However, this work only focuses on Denial of Service attacks in the general network.

## 3 Preliminaries

### *3.1 Reservoir Computing*

Among various types of reservoir computing (RC), the echo state network (ESN) is chosen as the underlying architecture to implement intrusion detection [29]. As shown in Fig. 1, an ESN comprises an input layer, a reservoir layer with $N$ nonlinear nodes, and an output layer. Mathematically, the reservoir layer and readout layer can be written as follows:

$$\boldsymbol{r}(t) = (1-a)\,\boldsymbol{r}(t-1) + a \cdot f\left(\boldsymbol{W} r(t-1) + \boldsymbol{W}_{in}\begin{bmatrix} b_{in} \\ \boldsymbol{u}(t) \end{bmatrix}\right) \tag{1}$$

$$\boldsymbol{y}(t) = \boldsymbol{W}_{out}\boldsymbol{R}(t) = \boldsymbol{W}_{out}\begin{bmatrix} b_{out} \\ \boldsymbol{u}(t) \\ \boldsymbol{r}(t) \end{bmatrix} \tag{2}$$

Here, $\boldsymbol{u}(t) \in \mathbb{R}^{Q}$ is the input vector collected at time $t = \{1, \ldots, T\}$, where $T$ is the number of data points in the training dataset, and $\boldsymbol{y}(t) \in \mathbb{R}^{P}$ is the output vector. $\boldsymbol{r}(t) \in \mathbb{R}^{N}$ is the state vector of $N$ reservoir nodes, which iterates as the input variable is updated. $\boldsymbol{W}_{in} \in \mathbb{R}^{N\times(1+Q)}$ maps an input to a high-dimensional space, uniformly and randomly distributed in $[-1, 1]$. $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is a random sparse matrix generated between $[0, 1]$, where the sparsity refers to the connectivity of the internal nodes of the reservoir layer. The spectral radius $\rho$ is the maximal absolute eigenvalue of $\boldsymbol{W}$ and must be scaled to

be no greater than 1, ensuring the echo state property of the ESN model [30]. The leakage parameter $a$, whose value ranges between [0, 1], represents the impact of the previous reservoir state on the current state. $\boldsymbol{W}_{out} \in \mathbb{R}^{Q\times(1+Q+N)}$ is the only matrix that needs to be trained between the reservoir and the output layer. The activation function $f(\cdot)$ used in the ESN is sigmoid or tanh, which introduces nonlinear factors to the nodes, allowing the RC to approximate arbitrary nonlinear functions and models. $b_{in}$ and $b_{out}$ are the input and output bias terms, respectively, both set as 1.

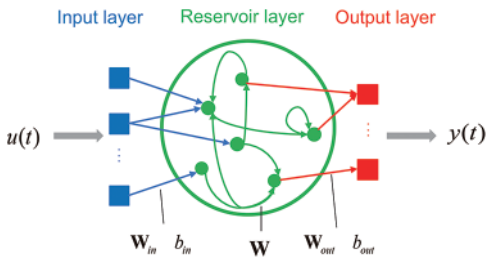


**Figure 1:** Principle structure diagram of ESN

### *3.2 Training Algorithm*

The computational efficiency of ESN lies in the fact that RC uses only one iteration to calculate the output weights, unlike gradient-based iterative optimization algorithms such as back-propagation which require multiple iterations to obtain optimal weights.

Instead of using the direct pseudoinverse, which is memory-intensive for large state-collecting matrices $R$ and limits the size of the reservoir $N$ and the number of training samples, ESN adopts the ridge regression algorithm, which can be expressed as follows:

$$\boldsymbol{W}_{out} = \boldsymbol{Y}\boldsymbol{R}^{T}\left(\boldsymbol{R}\boldsymbol{R}^{T} + \lambda\boldsymbol{I}\right)^{-1} \tag{3}$$

Here, $\boldsymbol{I} \in \mathbb{R}^{(1+Q+N)\times(1+Q+N)}$ is the identity matrix, $\lambda$ is the ridge regression parameter set as $10^{-8}$ to avoid overfitting and the reservoir nodes' state matrix $\boldsymbol{R} \in \mathbb{R}^{(1+Q+N)\times T}$ (respectively, limited observable target output vector $\boldsymbol{Y} \in \mathbb{R}^{P\times T}$) is the matrix whose $i$-th column is $[b_{out}; \boldsymbol{u}(i); \boldsymbol{r}(i)]$ (respectively, $[\boldsymbol{y}(i)]$).

## 4 The Proposed Intrusion Detection System

In this section, we introduce our innovative Intrusion Detection System (IDS) designed to safeguard both in-vehicle and external networks. Our IDS employs a two-fold approach, starting with the transformation of tabular data into images. Subsequently, we unveil the complete IDS scheme, leveraging the power of RC for robust intrusion detection.

### *4.1 Tabular Data to Image Transformation*

In the first phase of our IDS development, we focus on the critical process of converting tabular data into images. This transformation method enhances the interpretability and effectiveness of subsequent analysis, allowing for more advanced intrusion detection techniques.

We employ a transformation process that converts the raw tabular data into a structured image format. This transformation serves a fundamental purpose by harnessing principles from both computer vision and statistical data preprocessing. By representing the data as images, we leverage

the intrinsic human capacity for visual pattern recognition, enabling the application of visual analysis techniques.

Furthermore, as a crucial preprocessing step, we employ quantile normalization to rescale the network data into a standardized range of 0 to 255, a range commonly associated with pixel values in images. Quantile normalization is a robust statistical technique used to align the probability distribution of two datasets. Given two datasets, $X$ and $Y$, each containing $n$ observations, the process involves the following steps:

- **Sorting:** Sort both datasets in ascending order, denoted as $X_{sorted}$ and $Y_{sorted}$.
- **Rank Calculation:** Calculate the ranks of the observations in $X_{sorted}$, denoted as $R_X$, and similarly for $Y_{sorted}$, denoted as $R_Y$.
- **Quantile Values:** Compute the quantile values for $X_{sorted}$ and $Y_{sorted}$ as $Q_X = \frac{R_X}{n+1}$ and $Q_Y = \frac{R_Y}{n+1}$.
- **Mapping:** Map the quantile values of $X$ to those of $Y$ by finding a function $f$ such that $Q_Y = f(Q_X)$.
- **Normalization:** Apply the mapping function $f$ to the original dataset $X$ to obtain the quantile-normalized dataset $X_{normalized}$.

This process ensures that $X_{normalized}$ and $Y_{sorted}$ have the same quantile values, aligning their distributions. Quantile normalization is particularly useful for comparative analyses, such as microarray data, where it is crucial to remove systematic variations between datasets to make valid statistical inferences.

**Preposition 1.** *Let $X$ and $Y$ be two datasets with the same number of observations, $n$. After performing quantile normalization on $X$ and $Y$ according to the defined process, $X_{normalized}$ and $Y_{normalized}$ will have identical quantile values, thus aligning their probability distributions.*

*Proof*. Let $X_{sorted}$ and $Y_{sorted}$ represent the sorted versions of datasets $X$ and $Y$, respectively. The quantile values $Q_X$ and $Q_Y$ for $X_{sorted}$ and $Y_{sorted}$ are defined as:

$$Q_X = \frac{R_X}{n+1} \tag{4}$$

$$Q_Y = \frac{R_Y}{n+1} \tag{5}$$

where $R_X$ and $R_Y$ are the ranks of the observations in $X_{sorted}$ and $Y_{sorted}$, respectively.

During quantile normalization, we map the quantile values of $X$ to those of $Y$ using the function $f$, such that $Q_Y = f(Q_X)$. This ensures that the quantile values of $X_{normalized}$ and $Y_{normalized}$ are the same.

Therefore, by construction, the quantile normalization process guarantees that $X_{normalized}$ and $Y_{normalized}$ have identical quantile values, aligning their probability distributions.

Quantile normalization ensures that the resulting image representations capture the essential statistical properties of the data distribution while reducing the influence of outliers, which can be especially critical in real-world scenarios where network traffic data may exhibit variability and anomalies.

### *4.2 Wrapping Up: Malicious Classification Based on RC*

Building upon the tabular data-to-image transformation, we present a comprehensive IDS scheme that relies on RC. This scheme empowers us to tackle the complex task of classifying malicious activities with efficiency and precision, ensuring the security of in-vehicle and external networks.

The whole IDS scheme is outlined in Fig. 2. Initially, we collect data from the Internet of Vehicles (IoV) environment, followed by data pre-processing to improve data quality. Then, we convert vehicular tabular data into images to assist the RC model in identifying attack patterns in the data flow. As external vehicle data often suffers from class imbalance and feature redundancy issues, we utilize data resampling techniques such as $k$-means clustering and Synthetic Minority Oversampling Technique (SMOTE), along with feature engineering strategies based on information gain and correlation analysis. The next step involves labeling the generated images to form an image set, after which the RC model performs classification tasks to identify normal and attack behaviors. The details of this process are explained comprehensively in the subsequent subsections.

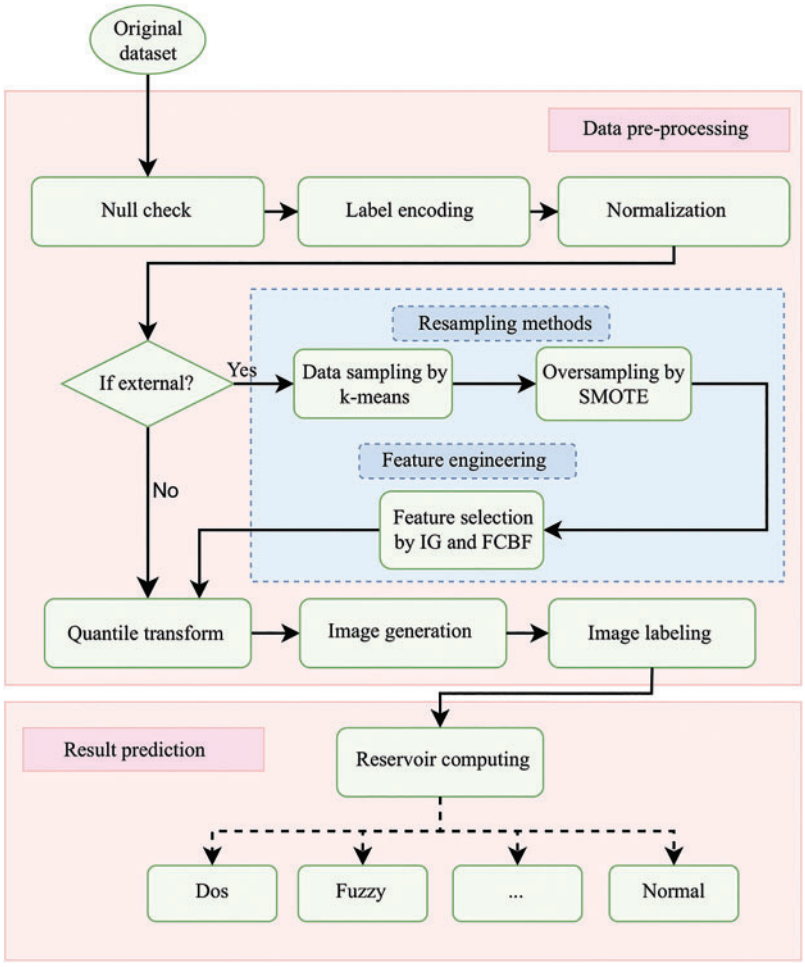


**Figure 2:** The proposed RC-based IDS framework

*4.2.1 Data Pre-Processing*

The data pre-processing phase begins with checking for missing values in the dataset. If any rows contain missing values, they are deleted to ensure data integrity. Next, we convert categorical features into numerical values using a label encoder, enabling direct processing by machine learning algorithms. After that, we apply Z-score normalization (ZN) to scale the data, which brings features with different ranges into a normalized range. Let $x_i (i = 1, 2, \ldots, D)$ denote the $i$-th component of each feature vector $\boldsymbol{x} \in \mathbb{R}^D$. We compute the mean $\mu_x$ and the standard deviation $\sigma_x$ of these $D$ components as follows:

$$\mu_x = \frac{1}{D} \sum_{i=1}^{D} x_i, \tag{6}$$

$$\sigma_x = \sqrt{\frac{1}{D} \sum_{i=1}^{D} (x_i - \mu_x)^2}. \tag{7}$$

ZN normalization is then applied to obtain the normalized data $\boldsymbol{x}^{(zn)}$ as follows:

$$\boldsymbol{x}^{(zn)} = ZN(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mu_x \mathbf{1}}{\sigma_x} \in \mathbb{R}^D. \tag{8}$$

Here, $\mathbf{1} = [1, 1, \ldots, 1]^T$ is a $D$-dimensional vector with all components being ones. This normalization ensures that each feature has a mean of 0 and a standard deviation of 1.

In cases where class imbalance is present, additional data pre-processing steps are required. Since external networks often generate a massive amount of data, it is unnecessary to spend significant time and resources training a machine learning model using redundant data. To address this, we employ the $k$-means clustering algorithm to reduce the data size and save training time. $k$-means clustering is a technique that groups similar entities within multiple subsets. The algorithm iteratively updates cluster centroids and boundaries to minimize the sum of squared distances from the data points to the corresponding cluster centers. We select representative subsets from each cluster and randomly discard redundant data.

The second step in combating skewed data distribution involves using the Synthetic Minority Over-sampling Technique (SMOTE) to generate substantial new artificial samples. In our case, the minority class refers to malicious attacks, which make up a negligible portion of the dataset and can result in a bias towards the normal class when training a model. SMOTE interpolates between randomly selected minority observations and their neighboring minority observations, creating new samples that are more representative of the minority class. The process can be described as follows: for a randomly chosen minority observation $a$, we select instance $b$ among its $k$-nearest minority class neighbors. To create a new sample, we interpolate between the two samples using a random weight $w$ ranging from 0 to 1. The new sample $j$ is generated as:

$$j = a + w \times (b - a) \tag{9}$$

*4.2.2 Feature Engineering*

Feature engineering aims to remove irrelevant or redundant features, thereby obtaining optimal feature subsets to enhance the performance of the subsequent classification algorithm. We employ

a two-step process for feature engineering: feature selection based on information gain and a fast correlation-based filter (FCBF).

- The information gain-based algorithm selects features based on the amount of information that can be gained from these features from an information theory perspective. By calculating entropy and mutual information, we obtain the subset of the most relevant features that contain the most information. The feature evaluation function is given by:
$$I(F|C) = H(F) - H(F|C) \tag{10}$$
where $I(F|C)$ is the mutual information between the feature subset $F$ and the class $C$, measuring the interdependence between them. $H(F)$ is the entropy of the discrete feature subset $F$. Given events $e_1, \ldots, e_m$ occurring with probabilities $p_1, \ldots, p_m$, the information entropy is defined as:
$$H = \sum_i p_i log \frac{1}{p_i} = -\sum_i p_i log p_i. \tag{11}$$
$H(F|C)$ is the conditional entropy of the discrete feature subset $F$, quantifying the uncertainty of $F$ given the class $C$. Mathematically, $H(F|C)$ is defined as:
$$H(F|C) = -\sum_{f \in F} \sum_{c \in C} p(f, c)\, logp(c|f) \tag{12}$$
where $p(f, c)$ represents the joint probability of $F$ taking the value $f$ and $C$ taking the value $c$, and $p(f|c)$ represents the conditional probability. Based on the ranking of the information gain values, we select the most important features that contain the most information.

- We further use the fast correlation-based filter to remove redundant features. Although we have selected the most relevant features, some unimportant features may still exist. The symmetrical uncertainty (SU) is computed to measure the correlation between features, and it is defined as follows:
$$SU(F, C) = 2 \times \left( \frac{I(F|C)}{H(F) + H(C)} \right) \tag{13}$$
A larger SU value indicates a higher correlation between the two given features, suggesting redundancy. In such cases, one of the redundant features needs to be removed.

Hereby we show that the combination of information gain-based feature selection and FCBF leads to a feature subset ($F_{FCBF}$) containing the most relevant information while removing redundancy.

**Preposition 2.** *Feature selection based on information gain and the fast correlation-based filter ensures that the selected feature subset contains the most relevant information while removing redundancy*.

*Proof*. To begin, notice that the information gain ($I(F_{IG}|C)$) is maximized during information gain-based feature selection. This ensures that the selected feature subset $F_{IG}$ contains the most relevant information for classifying instances.

Regarding FCBF for redundancy removal, notice that FCBF refines the feature subset $F_{IG}$ by maximizing symmetrical uncertainty ($SU(F_{IG}, C)$). This step ensures that features in $F_{FCBF}$ are highly correlated with the class ($C$) while minimizing inter-feature correlation (redundancy). Mathematically, FCBF selects features that satisfy:
$$\max_{F_{FCBF}} SU(F_{FCBF}, C) \tag{14}$$

Therefore, by maximizing I($F_{IG}$|C) and subsequently selecting $F_{FCBF}$ based on $SU(F_{FCBF}, C)$, we ensure that $F_{FCBF}$ contains the most relevant information while effectively removing redundancy among features. This completes the proof.

#### *4.2.3 Configuration*

The final step involves directly feeding the generated images into the ESN model without any additional feature extraction. This enables us to obtain a well-trained model capable of performing intrusion detection on test images. Specifically, we begin by constructing the ESN using randomly generated, but fixed, matrices $\boldsymbol{W}_{in}$ and $\boldsymbol{W}$. The training dataset, consisting of labeled images, is then input into the ESN for classification. Subsequently, the ESN is trained, and the matrix $\boldsymbol{W}_{out}$ is determined. Finally, the ESN, equipped with $\boldsymbol{W}_{in}$, $\boldsymbol{W}$, and the trained $\boldsymbol{W}_{out}$, is employed to classify vehicular traffic data into different categories.

As mentioned earlier in the ESN preliminaries, several paramters significantly impact the performance of ESN models, particularly the reservoir size $N$ and the leakage rate $a$. The reservoir size $N$ determines the ESN's ability to approximate complex transformations. However, increasing $N$ within a certain range improves performance at the cost of increased computational time. Beyond this range, further increases in $N$ no longer yield better performance (i.e., overfitting occurs), resulting in wasted computational resources. The leakage rate $a$ is related to the dynamics of reservoir updates. It represents the influence of previous states in the reservoir on the current state and also affects prediction performance to a certain extent. Therefore, we develop the optimal ESN model by adjusting the reservoir size $N$ and the leakage rate $a$.

### *4.3 Performance Analysis*

To justify the performance of Reservoir Computing (RC) in intrusion detection, we establish the Universal Approximation Theorem for Reservoir Computing (UAT-RC).

**Theorem 1.** *Let $X$ be a compact subset of $\mathbb{R}^n$, and $Y$ be a subset of $\mathbb{R}^m$. For any continuous function $g\colon X \to Y$, there exist suitable parameters for a reservoir computing system (e.g., an Echo State Network - ESN) such as the reservoir size N and the leakage rate a such that the system can approximate g with arbitrary accuracy. That is, for any $\epsilon > 0$, there exist reservoir parameters N and a and a readout layer such that the following holds:*

$$\|g(x) - \hat{g}(x)\| < \epsilon$$

*for all $x \in X$, where $g(x)$ is the true function value, $\hat{g}(x)$ is the value predicted by the reservoir computing system, and $\|\cdot\|$ denotes the L2 norm.*

*Proof: Step 1: Approximation of Continuous Functions.* First, we establish that a reservoir computing system, specifically an Echo State Network (ESN), can approximate continuous functions on a compact subset of $\mathbb{R}^n$ with arbitrary accuracy. For this purpose, consider any continuous function $g\colon X \to Y$, where $X$ is a compact subset of $\mathbb{R}^n$ and $Y$ is a subset of $\mathbb{R}^m$. By the Stone-Weierstrass Theorem, which states that the set of all polynomials is dense in the space of continuous functions, we know that for any $\epsilon > 0$ and any continuous functiong$\colon X \to Y$, there exists a polynomial $p(x)$ such that:

$$\|g(x) - p(x)\| < \frac{\epsilon}{2}$$

For all $x \in X$.

*Step 2: Approximation with Reservoir Computing*. Now, we aim to approximate the polynomial $p(x)$ using a reservoir computing system. Given the polynomial $p(x)$, we can set up an ESN with suitable parameters.

The ESN consists of a reservoir layer with $N$ nonlinear nodes, and its dynamics are governed by the following equation:

$$\boldsymbol{r}(t) = (1-a)\,\boldsymbol{r}(t-1) + a \cdot f\left(\boldsymbol{W} r(t-1) + \boldsymbol{W}_{in}\begin{bmatrix} b_{in} \\ \boldsymbol{u}(t) \end{bmatrix}\right)$$

where $\boldsymbol{r}(t)$ is the state vector, $\boldsymbol{W}_{in}$ and $\boldsymbol{W}$ are weight matrices, $a$ is the leakage parameter, and $f(\cdot)$ is the activation function. We can train the readout layer of the ESN, represented by $\boldsymbol{W}_{out}$, using a suitable algorithm (e.g., ridge regression) on the polynomial $p(x)$.

*Step 3: Achieving Arbitrary Accuracy*. By the properties of ESNs and the approximation properties of polynomials, we can choose the reservoir size $N$ and the leakage rate $a$ such that:

$$\|p(x) - \hat{g}(x)\| < \frac{\epsilon}{2}$$

where $\hat{g}(x)$ is the value predicted by the ESN-based system. Now, we can combine the errors from both steps:

$$\left\|g(x) - \hat{g}(x)\right\| \leq \|g(x) - p(x)\| + \left\|p(x) - \hat{g}(x)\right\| < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

This proves that for any continuous function $g(x)$, there exist suitable parameters for an ESN, including $N$ and $a$, and a readout layer such that the ESN can approximate $g(x)$ with an error less than $\epsilon$.

The above theorem demonstrates that ESNs can approximate continuous functions on a compact subset of $\mathbb{R}^n$ with arbitrary accuracy. This demonstrates the effectiveness of Reservoir Computing in approximating complex relationships in network traffic data, justifying its utility in intrusion detection.

## 5 Simulation Results

The experiments were conducted on a Windows 10 64-bit operating system using a Python 3.9 environment running on an Intel Core i7-10700 CPU (2.90 GHz). We will compare RC-based IDS with the most relevant and representative neural networks-based methods in the Related Work section. The research focus of [27] and [28] also includes privacy-preserving techniques based on blockchain, therefore, the benchmark algorithms employed for comparison were the following: 1-dimensional CNN (1DCNN) [11], deep convolutional neural network (DCNN) [12], feed-forward neural network [13] (FFNN), and transfer learning and optimized CNN-based IDS [14], referred to as CNN (Concatenation) and CNN (Confidence Averaging).

### *5.1 Settings*

We utilize two benchmark datasets to represent in-vehicle and external networks: the Car-Hacking dataset and the CICIDS2017 dataset, respectively. The Car-Hacking dataset was constructed by capturing Controller Area Network (CAN) traffic logs from an actual vehicle. Table 1 provides details of the 5% Car-Hacking dataset, which consists of four attacks: Denial of Service (DoS), Fuzzy, Gear Spoofing, and RPM Spoofing. The dataset includes the CAN ID and the 8-bit data of CAN packets (DATA [0]-DATA [7]). On the other hand, the CICIDS2017 dataset contains normal and common attack behaviors. It consists of 2,830,743 rows, with each row consisting of 79 features and labeled as

either benign or one of 14 attack types. The attacks can be categorized into five main types based on [31]. Table 2 presents the distribution of the different attack types and the number of benign rows in the CICIDS2017 dataset.

**Table 1:** Details of the 5% Car-Hacking dataset

| Class label | Number of samples |
|---|---|
| Normal | 701832 |
| Dos | 29501 |
| Fuzzy | 24624 |
| RPM spoofing | 32539 |
| Gear spoofing | 29944 |

**Table 2:** Details of the CICIDS2017 dataset

| Class label | Attack type | Number of samples |
|---|---|---|
| Benign | – | 2273097 |
| Bot | Botnet | 1966 |
| DDos | Dos | 380699 |
| Dos golden eye | | |
| Dos hulk | | |
| Dos-httptest | | |
| Dos-slowloris | | |
| Heartbleed | | |
| Port-scan | Sniffing | 158930 |
| SSH-patator | Brute-Force | 13835 |
| FTP-patator | | |
| Infiltration | Infiltration | 36 |
| Web attack-brute force | Web attack | 2180 |
| Web attack-sql injection | | |
| Web attack-XSS | | |

The normalized tabular data is divided into chunks based on timestamps and feature sizes to exploit the temporal correlation of traffic data for the Recurrent Classifier (RC). For the Car-Hacking dataset, we select 27 consecutive samples to form an image with a size of $9 \times 9 \times 3$. This dataset has 9 features, and the resulting image has 3 channels. For the CICIDS2017 dataset, we apply feature engineering to select the most relevant 20 features. Therefore, the transformed image size of the CICIDS2017 dataset is $20 \times 20 \times 3$, based on $20 \times 3$ consecutive data rows. The labels of the images are assigned based on the highest proportion of attack types present. If an image contains no malicious attacks, it is labeled as "normal". The shuffled datasets are then split into two parts, with 80% of the images allocated to the training dataset and the remaining 20% to the test dataset.

The evaluation metrics utilized in the analysis include accuracy, precision, recall, and F1-scores, which are calculated based on the elements of the confusion matrix: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The definitions of these metrics are as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{18}$$

The training time of each method was also recorded.

### *5.2 Performance Analysis*

The comparison results are presented in Tables 3 and 4. Multiple experiments were conducted to determine the optimal parameters for the Echo State Network (ESN). In the notation ESN-$A$-$N$, $A$ represents the leakage rate, and $N$ denotes the number of reservoir nodes. For the Car-Hacking dataset, the number of reservoir nodes was fixed at 100, while for the CICIDS2017 dataset, it was set to 1000 due to the increased complexity of the attack patterns requiring more neurons for detection. In the experimentation process, the leakage rate was varied from 0.2 to 0.8 with an interval of 0.2. After identifying the best leakage rate for the Car-Hacking dataset, the number of nodes was gradually increased to strike a balance between performance and training time.

The training time of the ESN model consists of two parts: the iteration time for updating the reservoir state and the time for ridge regression.

**Table 3:** Performance comparisons on the Car-Hacking dataset

| Method | Accuracy (%) | Precision (%) | Recall (%) | F1-score (%) | Training time (s) |
|---|---|---|---|---|---|
| 1DCNN [11] | 99.96 | 99.94 | 99.63 | 99.80 | N/A |
| DCNN [12] | 99.93 | 99.84 | 99.84 | 99.71 | N/A |
| CNN (Concatenation) [14] | 100 | 100 | 100 | 100 | 2490.5 |
| CNN (Confidence averaging) [14] | 100 | 100 | 100 | 100 | 1680.7 |
| ESN-0.2-100 | 98.08 | 96.47 | 88.98 | 91.09 | 14.3 |
| ESN-0.4-100 | 99.48 | 97.92 | 97.02 | 97.32 | 13.6 |
| ESN-0.6-100 | 99.88 | 99.38 | 99.31 | 99.34 | 13.9 |
| ESN-0.8-100 | 99.86 | 99.38 | 99.20 | 99.30 | 13.5 |
| ESN-0.6-200 | 100 | 100 | 100 | 100 | 14.6 |
| ESN-0.6-300 | 100 | 100 | 100 | 100 | 15.3 |
| ESN-0.6-500 | 100 | 100 | 100 | 100 | 16.3 |

**Table 4:** Performance comparisons on the CICIDS2017 dataset

| Method | Accuracy (%) | Precision (%) | Recall (%) | F1-score (%) | Training time (s) |
|---|---|---|---|---|---|
| FFNN [13] | 99.46 | 99.52 | 99.40 | 99.46 | N/A |
| CNN (Concatenation) [14] | 99.899 | 99.900 | 99.899 | 99.898 | 3598.7 |
| CNN (Confidence averaging) [14] | 99.925 | 99.825 | 99.824 | 99.925 | 2658.1 |
| ESN-0.1-1000 | 91.21 | 55.16 | 50.68 | 52.10 | 35.9 |
| ESN-0.2-1000 | 92.52 | 55.51 | 52.01 | 53.26 | 35.6 |
| ESN-0.4-1000 | 93.10 | 64.97 | 61.42 | 62.75 | 35.2 |
| ESN-0.6-1000 | 92.49 | 63.03 | 61.22 | 61.54 | 35.9 |
| ESN-0.8-1000 | 91.79 | 55.70 | 51.17 | 52.64 | 36.0 |
| ESN-0.4-2000 | 94.47 | 79.31 | 64.57 | 67.07 | 377.4 |
| ESN-0.4-3000 | 96.63 | 85.07 | 73.71 | 78.40 | 1038.2 |
| ESN-0.4-4000 | 97.79 | 95.28 | 88.86 | 87.76 | 1879.0 |
| ESN-0.4-5000 | 98.47 | 99.62 | 98.17 | 96.47 | 2531.0 |

From the results presented in Table 3 for the Car-Hacking dataset, it can be observed that when the number of reservoir nodes is fixed at 100, the best performance is achieved with a leakage rate of 0.6. As the number of nodes ($N$) increases, both the performance and training time increase. However, when the number of nodes reaches 200, the performance becomes saturated. It is important to note that while both ESN and CNN achieve 100% accuracy, precision, recall, and F1-scores, the training time of ESN is significantly faster compared to CNN-based methods.

When analyzing the CICIDS2017 dataset (Table 4), it can be observed that the best leakage rate for this dataset is 0.4. The ESN-0.4-5000 method achieves an accuracy of 98.47% and a precision of 99.62% while only requiring 2531.0 s of training time. The performance of ESN on the CICIDS2017 dataset is slightly less significant compared to other methods, but it still demonstrates comparable performance while offering advantages in terms of training time and storage size.

ESN also has a significant advantage in terms of model storage. The number of well-trained parameters, represented by the readout layer weights, for ESN is only $N_{att} \times (N + L + 1)$, where $N_{att}$ is the number of attack types (e.g., $N_{att} = 6$ in CICIDS2017), and $L$ is the image length. This is in contrast to traditional CNN models such as VGG16, which have trained ImageNet weights of approximately 528 MB.

## 6 Conclusions

In this study, we proposed an RC-based IDS framework to protect IoV systems against intra-vehicle and external network attacks. This research marks the first example of applying RC in IoV intrusion detection, effectively mitigating the computational resource and memory space constraints associated with traditional machine learning models. The data preprocessing steps, including resampling, feature engineering, and data transformation, make this RC-based IDS framework more robust. We conducted experiments using both the Car-Hacking dataset and the CICIDS2017 dataset. The optimal number of nodes and leakage rate in the ESN were explored. The results indicate that our

proposed framework performs well on intra-vehicle datasets with minimal training time. Although the superiority of the CICIDS2017 dataset is less significant, the comparable performance demonstrates lightweight properties in terms of training time and storage size. In future work, we plan to apply Reservoir Computing (RC) to detect zero-day attacks, aiming to further enhance the security of IoV.

**Acknowledgement:** The authors appreciate the valuable comments from the reviewers.

**Funding Statement:** This work was supported in part by the Open Research Fund of Joint Laboratory on Cyberspace Security, China Southern Power Grid (Grant No. CSS2022KF03), and the Science and Technology Planning Project of Guangzhou, China (Grant No. 202201010388), the Fundamental Research Funds for the Central Universities.

**Author Contributions:** The authors confirm contribution to the paper as follows: original draft preparation: Jiarui Deng; draft review and edition: Wuqiang Shen, Yihua Feng, Guosheng Lu; conceptualization supervision and revision: Shanxiang Lyu. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The Car-Hacking and CICIDS2017 datasets used in this paper can be accessed at https://ocslab.hksecurity.net/Datasets/car-hacking-dataset and https://www.unb.ca/cic/datasets/ids-2017.html, respectively.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[2] J. Contreras-Castillo, S. Zeadally and J. A. Guerrero-Ibañez, "Internet of vehicles: Architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[3] B. Ji, X. Zhang, S. Mumtaz, C. Han, C. Li *et al.,* "Survey on the internet of vehicles: Network architectures and applications," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 34–41, 2020.

[4] C. Zhang, X. Luo, J. Liang, X. Liu, L. Zhu *et al.,* "POTA: Privacy-preserving online multi-task assignment with path planning," *IEEE Transactions on Mobile Computing,* pp. 1–13. IEEE. https://doi.org/10.1109/TMC.2023.3315324

[5] C. Zhang, C. Hu, T. Wu, L. Zhu and X. Liu, "Achieving efficient and privacy-preserving neural network training and prediction in cloud environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4245–4257, 2023.

[6] B. Zhao, J. Yuan, X. Liu, Y. Wu, H. H. Pang *et al.,* "SOCI: A toolkit for secure outsourced computation on integers," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3637–3648, 2022.

[7] B. Zhao, X. Liu, A. Song, W. N. Chen and K. K. Lai, "PriMPSO: A privacy-preserving multiagent particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, vol. 53, no. 11, pp. 7136–7149, 2023.

[8] C. Hu, C. Zhang, D. Lei, T. Wu, X. Liu *et al.,* "Achieving privacy-preserving and verifiable support vector machine training in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3476–3491, 2023.

[9] S. Sharma and B. Kaushik, "A survey on internet of vehicles: Applications, security issues & solutions," *Vehicular Communications*, vol. 20, pp. 100182, 2019.

[10] L. Malina, P. Seda, Z. Martinasek, J. Pokorný, M. Srotyr *et al.,* "On security and privacy in vehicle speed-limiting services in the internet of vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 15, no. 1, pp. 8–22, 2023.

[11] 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 *IEEE Global Communications Conf., GLOBECOM 2020*, Taiwan, pp. 1–6, 2020.
[12] H. M. Song, J. Woo and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, pp. 100198, 2020.
[13] A. Rosay, F. Carlier and P. Leroux, "Feed-forward neural network for network intrusion detection," in *91st IEEE Vehicular Technology Conf., VTC Spring 2020*, Antwerp, Belgium, pp. 1–6, 2020.
[14] L. Yang and A. Shami, "A transfer learning and optimized CNN based intrusion detection system for internet of vehicles," in *IEEE Int. Conf. on Communications, ICC 2022*, Seoul, Korea, pp. 2774–2779, 2022.
[15] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng *et al.,* "Data-driven intrusion detection for intelligent internet of vehicles: A deep convolutional neural network-based method," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2219–2230, 2020.
[16] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.
[17] K. Bekshentayeva and L. Trajkovic, "Detection of denial of service attacks using echo state networks," in *2021 IEEE Int. Conf. on Systems, Man, and Cybernetics, SMC 2021*, Melbourne, Australia, pp. 1227–1232, 2021.
[18] K. Hamedani, L. Liu, R. Atat, J. Wu and Y. Yi, "Reservoir computing meets smart grids: Attack detection using delayed feedback networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 734–743, 2018.
[19] E. Seo, H. M. Song and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *16th Annual Conf. on Privacy, Security and Trust, PST 2018*, Belfast, Northern Ireland, UK, pp. 1–6, 2018.
[20] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of the 4th Int. Conf. on Information Systems Security and Privacy, ICISSP 2018*, Funchal, Madeira-Portugal, pp. 108–116, 2018.
[21] A. R. Gad, A. A. Nashat and T. M. Barkat, "Intrusion detection system using machine learning for vehicular ad hoc networks based on ToN-IoT dataset," *IEEE Access*, vol. 9, pp. 142206–142217, 2021.
[22] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo *et al.,* "A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network," *IEEE Access*, vol. 7, pp. 154560–154571, 2019.
[23] L. Yang, A. Moubayed, I. Hamieh and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *2019 IEEE Global Communications Conf. (GLOBECOM)*, Hawaii, USA, pp. 1–6, 2019.
[24] F. Jin, M. Chen, W. Zhang, Y. Yuan and S. Wang, "Intrusion detection on internet of vehicles via combining log-ratio oversampling, outlier detection and metric learning," *Information Sciences*, vol. 579, pp. 814–831, 2021.
[25] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi and A. Strelzoff, "An explainable deep learning framework for resilient intrusion detection in IoT-enabled transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1000–1014, 2022.
[26] S. Anbalagan, G. Raja, S. Gurumoorthy, R. D. Suresh and K. Dev, "IIDS: Intelligent intrusion detection system for sustainable development in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15866–15875, 2023.
[27] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta and N. Kumar, "P2SF-IoV: A privacy-preservation-based secured framework for internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22571–22582, 2022.
[28] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, N. Kumar *et al.,* "A privacy-preserving-based secure framework using blockchain-enabled deep-learning in cooperative intelligent transport system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16492–16503, 2022.

[29] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," *GMD Technical Report*, vol. 148, no. 34, pp. 13, 2001.
[30] M. Lukosevicius, "A practical guide to applying echo state networks, in neural networks: Tricks of the trade," In: G. Montavon, G. B. Orr, K. R. Müller (Eds.), *Neural Networks: Tricks of the Trade-Second Edition*, vol. 7700, pp. 659–686, Berlin, Heidelberg: Springer, 2012.
[31] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3, pp. 479–482, 2018.