electronics

MDPI

Article

A New DSGRU-Based Intrusion Detection Method for the Internet of Things

Yueling Liu [1], Yingcong Lan [1], Changsong Yang [1,2,3,*], Yong Ding [1,4] and Chunhai Li [1]

[1] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China; ylliu@guet.edu.cn (Y.L.); 22032201048@mails.guet.edu.cn (Y.L.); stone_dingy@guet.edu.cn (Y.D.); chunhaili@guet.edu.cn (C.L.)
[2] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518055, China
[3] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
[4] Institute of Cyberspace Technology, HKCT Institute for Higher Education, Hong Kong 999077, China
* Correspondence: csyang@guet.edu.cn; Tel.: +86-13397838265

Abstract: The Internet of Things (IoT), a rapidly developing technology, connects entities to the Internet through information sensing devices and networks. Recently, IoT has gained widespread application in daily life and work due to its high efficiency and convenience. However, with the rapid development of IoT, the systems are intruded upon by malicious users and hackers more and more frequently. As a result, intrusion detection has attracted significant attention, and numerous schemes have been proposed that can precisely identify malicious intrusion operations. However, the existing schemes suffer from several severe challenges, such as low accuracy, high computational overhead, and poor real-time performance, in processing large-scale, high-dimensional, and temporally correlated IoT network traffic data. To address these challenges, we propose a new intrusion detection scheme for IoT in this paper. Specifically, we first improve the traditional Gate Recurrent Unit (GRU) and design a novel neural network model, namely, the Deep Supplement Gate Recurrent Unit (DSGRU). This model comprises an Original Gate Recurrent Unit (OGRU), a Decode Gate Recurrent Unit (DGRU), and a Softmax activation function. Compared with the traditional GRU, our proposed DSGRU can more efficiently extract features from IoT network traffic data and reduce the loss of features caused by nonlinear transformations during the learning process. Subsequently, we adopt DSGRU to design a novel intrusion detection scheme for IoT. We also analyze the theoretical computational complexity of the proposed scheme. Finally, we implement our proposed intrusion detection scheme and evaluate its performance based on the UNSW-NB15 and NSL-KDD datasets. The experimental results demonstrate that our proposed DSGRU-based intrusion detection scheme achieves better performance, including in terms of *Accuracy*, *Precision*, *Recall*, *F1_score*, loss value, and efficiency.

Keywords: network security; Internet of Things; intrusion detection; deep learning; deep supplement gate recurrent unit

Citation: Liu, Y.; Lan, Y.; Yang, C.; Ding, Y.; Li, C. A New DSGRU-Based Intrusion Detection Method for the Internet of Things. *Electronics* **2023**, *12*, 4745. https://doi.org/10.3390/electronics12234745

Academic Editor: Lei Shu

Received: 17 October 2023
Revised: 19 November 2023
Accepted: 20 November 2023
Published: 23 November 2023

1. Introduction

The IoT, a rapidly developing and promising technology, connects entities to the internet through information sensing devices (such as infrared sensors, laser scanners, global positioning systems, and others) and a communication network based on agreed-upon protocols, thus forming a network of everything [1–3]. Due to its attractive advantages of high efficiency and convenience, IoT has been widely applied in daily life and work. A report by IoT Analytics showed that the enterprise IoT market reached USD 201 billion at the end of 2022, and will reach USD 483 billion by the end of 2027. However, alongside the rapid development and widespread application of IoT, it is increasingly targeted by hackers, and its security has become a major concern for users [4].

To protect IoT security, plenty of security protection technologies have been applied in the IoT landscape, such as firewalls [5,6], authentication mechanisms [7,8], data encryption [9–11], and so on. Although these technologies can prevent many attacks, they are all passive defenses. That is, they can only protect IoT security when attacks occur. Meanwhile, they lack the capability to actively analyze network behaviors, identify threat patterns, or forecast attacks, thus limiting their effectiveness against a wide array of network attacks [10]. To achieve active defense, intrusion detection systems have been deployed in IoT. By collecting and analyzing network traffic data, these systems can determine whether there are malicious intrusions [12]. Upon detecting intrusions, they can also promptly intercept and respond to the attacks. Additionally, they can inform system administrators of malicious intrusions, thereby assisting them in taking related measures [13].

Recently, thanks to the rapid advancement of artificial intelligence, machine learning and deep learning have been widely applied in intrusion detection, resulting in a substantial body of literature. However, these schemes cannot be directly suitable for IoT because IoT network traffic data are characterized by large-scale, heterogeneous multiple sources, high dimensionality, and temporal correlation [14]. According to a report by IoT Analytics, the number of devices accessing the IoT is expected to reach 22 billion globally by the end of 2025 [2]. Generally, these IoT devices are constantly collecting and sharing data, thus generating large-scale, heterogeneous, high-dimensional, and temporally correlated network traffic data [15,16]. The existing intrusion detection solutions directly applied in the IoT environment will suffer from low accuracy, high computational overhead, and poor real-time performance.

Specifically, in the data processing phase, the existing schemes require a significant computational overhead to process large-scale IoT network traffic data, thus incurring substantial time overhead. As a result, this can greatly affect the overall efficiency of intrusion detection [17]. Meanwhile, to learn and characterize the characteristics of IoT network traffic data, these schemes necessitate numerous matrix calculations and nonlinear transformations, which might lead to feature loss. These issues can diminish the capacity for data characterization and might impair the model's ability to generalize, potentially reducing its applicability and universality. Therefore, the motivation of this paper is to design a novel intrusion detection scheme for IoT, one that can effectively process large-scale, heterogeneous, high-dimensional, and temporally correlated IoT network traffic data.

1.1. Contributions

In this paper, we design a new neural network model and use it to propose an intrusion detection scheme for IoT, which can achieve efficient and accurate intrusion detection. As a consequence, the main contributions of this paper can be summarized as follows:

- We improve the traditional GRU model and design a novel neural network algorithm, namely, DSGRU, which consists of OGRU, DGRU, and the Softmax activation function. Specifically, OGRU is utilized to characterize the learned network traffic data, while DGRU is utilized to decode and recover the features that have been characterized by OGRU. Meanwhile, DSGRU can better characterize the network traffic data and discover the intrinsic connections within the temporally correlated network traffic data. Moreover, DSGRU can greatly reduce feature loss during the nonlinear transformation process through loss compensation, thus effectively enhancing the robustness and generalization ability.
- We adopt DSGRU to propose a novel intrusion detection scheme for the IoT environment, which is ideally suited for large-scale IoT network traffic data characterized by heterogeneous multiple sources, high dimensionality, and temporal correlation. By employing a loss compensation approach, our proposed scheme can reduce feature loss of IoT network traffic data during the model training process. Meanwhile, our scheme can effectively learn the temporal correlation between IoT network traffic data, thus improving intrusion detection accuracy. Moreover, our scheme enhances efficiency by reducing the operations involved in nonlinear transformations.

- We provide the total computational time overhead and theoretical computational complexity analysis to demonstrate the efficiency of our proposed scheme. Meanwhile, we develop a prototype implementation and simulate our proposed scheme. Finally, we test our proposed scheme based on the UNSW-NB15 dataset and NSL-KDD dataset, and provide the overall performance evaluation. The experimental results not only demonstrate that our proposed scheme is suitable for the IoT environment but also demonstrate the practicability and universality of our proposed scheme.

The structure of this article is as follows. Some preliminaries will be described in Section 2. In Section 3, we propose a novel neural network model that will be utilized to establish our intrusion detection scheme. In Section 4, we first describe the system framework and then present the detailed intrusion detection scheme for the IoT environment. We provide the theoretical computational complexity comparison in Section 5. In Section 6, we implement our proposed scheme and provide the performance evaluation. Finally, we present our discussion and conclusion in Sections 7 and 8, respectively.

1.2. Related Works

Intrusion detection was first proposed by Denning in 1987 [18], which provided a general framework for intrusion detection. After that, it has attracted significant attention from both academia and industry, leading to a series of solutions. Generally, over the last decade, probabilistic statistics and deep learning have been widely utilized to achieve intrusion detection.

A lot of the existing solutions utilize classical data classification methods from machine learning to detect intrusion traffic data [19,20]. To detect IoT cyber attacks, Kuang et al. [21] combined Kernel Principal Component Analysis (KPCA) with Improved Chaotic Particle Swarm Optimization (ICPSO) to propose a new Support Vector Machine (SVM) model. Subsequently, they used this new SVM for intrusion detection. By integrating SVM and plain Bayesian feature embedding, Gu et al. [22] designed a new intrusion detection scheme. In their solution, they first performed a plain Bayesian feature transformation based on the original data features. Subsequently, they trained a model based on the data obtained after the transformation and utilized the trained model to reach intrusion detection. By combining an artificial neural network and SVM, Hussain et al. [23] designed a two-stage hybrid intrusion detection scheme. In their scheme, they utilized SVM to achieve anomaly detection and adopted an artificial neural network to complete misuse detection. Lv et al. [24] designed a hybrid kernel function for a limited learning machine and utilized it to establish an intrusion detection scheme. Additionally, they achieved automatic parameter optimization by combining a novel Gravitational Search Algorithm (GSA) with the Differential Evolution (DE) algorithm. Saleh et al. [25] achieved intrusion detection based on an optimized k-nearest neighbor and SVM classifier. However, traditional classification techniques cannot realize automatic parameter optimization during the data pre-processing and feature extraction phases. That is, they require reliance on the expertise of specialists and achieve learning and classification through human involvement. Therefore, they are not able to quickly and accurately process large-scale, heterogeneous, high-dimensional, and temporally correlated IoT network traffic data.

Deep learning, a novel research hotspot in neural networks and artificial intelligence, has been extensively applied in network intrusion detection and has achieved outstanding results [26,27]. In 2018, to satisfy the security requirements of the modern network, Shone et al. [28] designed a novel intrusion detection scheme based on a stacked Nonsymmetric Deep Autoencoder (NDAE). Experimental results demonstrated the strong potential of the proposal in modern networks. To achieve intrusion detection in wireless sensor networks, Otoum et al. [29] developed an intrusion detection system based on clustered constrained Boltzmann machines. In 2020, Kasongo and Sun [30] investigated the design of an intrusion detection scheme in a wireless network environment. Subsequently, by combining a Feed-Forward Deep Neural Network (FFDNN) with a Wrapper-Based Feature Extraction Unit (WFEU), they proposed a network intrusion detection scheme,

which could improve effectiveness and accuracy. To protect the security of industrial Cyber-Physical Systems (CPSs), Li et al. [31] designed an intrusion detection scheme based on a Convolutional Neural Network (CNN) and GRU. Almutlaq et al. [32] proposed a two-stage intrusion detection method for an intelligent transportation system, which could effectively improve the accuracy and efficiency of intrusion detection.

To improve the performance of intrusion detection in distributed networks, Keshk et al. [33] proposed a new SPIP IoT intrusion detection framework to identify network attacks and explain the model's decisions. Wu et al. [34] presented a hybrid intrusion detection scheme based on Stacked Autoencoders (SAEs) and the Kernel Approximation algorithm. They implemented their proposed scheme and provided a performance evaluation based on the NSL-KDD dataset. The experimental results verify the efficiency and accuracy of their proposal. To achieve intrusion detection in imbalanced datasets, Lee et al. [35] proposed a new intrusion detection method by combining an Autoencoder (AE) with a Generative Adversarial Network (GAN). Specifically, they used GAN to sample a few classes of data and adopted AE to downsize the data, thus improving overall performance. Lo et al. [36] proposed a Long Short-Term Memory (LSTM) algorithm, and then utilized LSTM to establish a new intrusion detection system with a high false alarm rate.

To detect the unbalanced attack types, Imrana et al. [37] proposed a Bidirectional LSTM for intrusion detection and the experiment results showed the effectiveness of Bidirectional LSTM. Aljrees et al. [38] introduced a novel paradigm that synergized efficient data encryption, the Quondam Signature Algorithm (QSA), and federated learning, to effectively detect random attacks in the IoT system. Sook et al. [39] proposed a two-phase network traffic anomaly detection system compatible with the ETSI-NFV standard 5G architecture. Although the above solutions can improve the accuracy of intrusion detection, they need to perform a large number of nonlinear transformations when dealing with large-scale, heterogeneous, high-dimensional, and temporally correlated IoT network traffic data, thus affecting efficiency and real-time performance. Meanwhile, they suffer from the problem of local data feature loss, which can reduce accuracy and generalization ability [40].

2. Preliminaries

To solve the problems of ineffective long-term memory information transfer and gradient vanishing in backpropagation, a new recurrent neural network GRU, which is very similar to the Long Short-Term Memory (LSTM) neural network, has been widely used [41]. GRU can efficiently capture the semantic associations between long sequences, thus mitigating the problems of gradient vanishing and gradient explosion. However, the structure and computation of GRU are both simpler than those of LSTM. GRU mainly consists of two gate structure units: the update gate and the reset gate, as demonstrated in Figure 1.

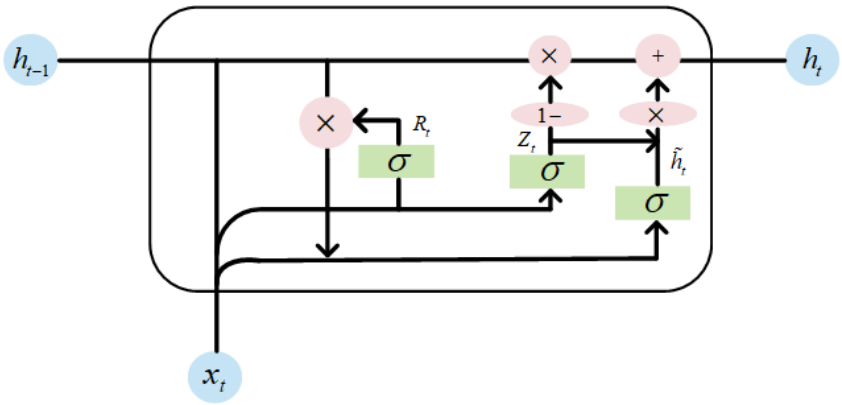

Figure 1. The structure of the GRU model.

In GRU, the reset gate controls the flow of the previous moment's hidden state into the candidate's hidden state of the current moment, which may contain all the historical

information about the time series up to that previous moment. Therefore, the reset gate enables the discarding of historical information that is not relevant to the prediction. The update gate controls the extent to which the current moment forgets the content of messages unrelated to the current moment and determines how much of the current candidate's hidden state is retained.

As shown in Figure 1 symbol R_t denotes the reset gate and symbol Z_t denotes the update gate. The GRU model can be described as follows.

Firstly, the reset gate R_t and the update gate Z_t are jointly determined by the past-moment information t and the current-moment information x_t, where x_t is a small batch of inputs at a given moment t. Subsequently, the reset gate R_t and the update gate Z_t can be computed as follows:

$$R_t = \sigma(W_R \cdot [h_{t-1}, x_t]) \tag{1}$$

$$Z_t = \sigma(W_Z \cdot [h_{t-1}, x_t]) \tag{2}$$

where σ is the sigmoid function, and W_R and W_Z are learnable parameters.

Secondly, the candidate's hidden state $\tilde{h}_t$ of GRU at moment t is controlled by the information of the past moment h_{t-1} and current moment x_t. The computational formula is described as follows:

$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [R_t \times h_{t-1}, x_t]) \tag{3}$$

where tanh is the tanh activation function and $W_{\tilde{h}}$ is the learnable parameter.

Finally, when calculating the hidden state h_t at moment t, it is necessary to utilize the update gate Z_t at the current moment to combine the hidden state h_{t-1} and the candidate's hidden state $\tilde{h}_t$. The specific formula is as follows:

$$h_t = (1 - Z_t) \times h_{t-1} + Z_t \times \tilde{h} \tag{4}$$

3. Deep Supplement Gate Recurrent Unit

Although GRU can achieve impressive performance in processing temporally correlated network traffic data, it requires performing numerous nonlinear transformations during the data feature learning process, resulting in feature loss and reduced efficiency. To address these issues, we have improved upon the traditional GRU and designed a novel neural network model called DSGRU. Its construction is demonstrated in Figure 2.

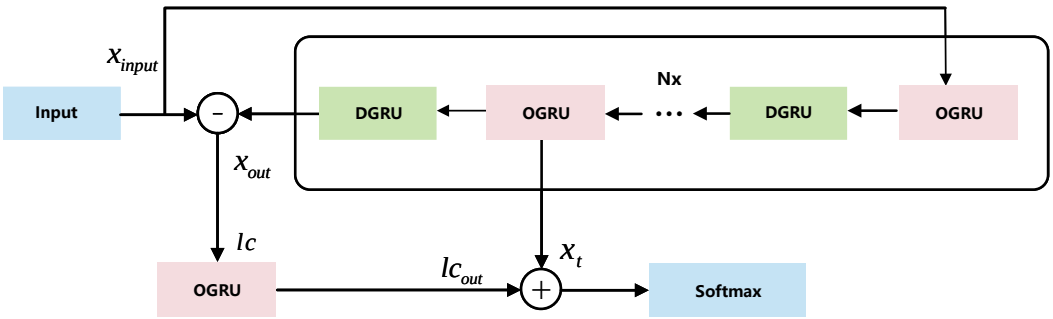

Figure 2. The structure of the DSGRU model.

We can directly see from Figure 2 that the proposed DSGRU consists of two types of GRU, i.e., OGRU and DGRU. Generally, OGRU contains a traditional GRU network and a fully connected layer. Meanwhile, OGRU is primarily utilized to learn data characterization from the input data. Similarly, DGRU consists of a traditional GRU and a fully connected layer, but its main function is to decode and recover the feature data after OGRU has learned the data characterization.

Unlike the traditional GRU, DSGRU is equipped with more hidden layers by deepening the OGRUs and the DGRUs. The deeper OGRUs and DGRUs can learn data features more fully, recover more data features, and lose fewer data features, thus improving the

learning characterization ability. At the same time, by increasing the layers of DGRU, it can retain more original data features in data decoding and provide more accurate loss compensation data in subsequent loss compensation. Subsequently, the overall data feature information loss is reduced through accurate loss compensation

In Figure 2, we utilize the symbol x_{input} to represent the input data and x_t to represent the data characterization of the output of OGRU located in the last layer. Moreover, we use x_{out} to denote the decoded data output of the last layer of DGRU, lc to denote the loss data, and lc_{out} to denote the loss compensation data. N_x denotes the number of layers of encoding and decoding. For simplicity, we use $N_x = 2$ as an example to illustrate a specific flow of DSGRU in the following part.

Firstly, x_{input} is assigned to x_{out}, and then x_{out} is encoded and decoded twice. Subsequently, x_t is obtained by the characterization learning executed by the OGRU located in the last layer. At the same time, x_{out} is obtained by the last layer of DGRU decoding, which can be calculated as follows:

$$x_{out} = x_{input} \tag{5}$$

$$x_t = OGRU(x_{out}) \tag{6}$$

$$x_{out} = DGRU(x_t) \tag{7}$$

$$x_t = OGRU(x_{out}) \tag{8}$$

$$x_{out} = DGRU(x_t) \tag{9}$$

Secondly, the loss data, lc, are obtained by subtracting the input data, x_{input}, from the x_{out} data obtained by the DGRU. Therefore, the loss data, l_c, can be computed by using the following formula:

$$lc = x_{input} - x_{out} \tag{10}$$

Finally, the loss compensation data, lc_{out}, are obtained through OGRU representation learning; then, x_t plus lc_{out} are assigned to x_t, and the final result is obtained after a Softmax activation function with the following formula:

$$lc_{out} = OGRU(lc) \tag{11}$$

$$x_t = x_t + lc_{out} \tag{12}$$

4. Our Proposed Scheme

In the following section, we will first describe the system framework of our new intrusion detection scheme. Subsequently, we will introduce the DSGRU-based intrusion detection scheme in detail.

4.1. System Framework

We investigate the design of an intrusion detection scheme for IoT. Recently, with the rapid advancement of information technology, especially in cloud computing, wireless networks, and sensor technology, the popularity and coverage of IoT have progressively increased. At the same time, the practical application of intrusion detection often involves network traffic data characterized by large-scale, heterogeneous multi-source, high dimensionality, and time correlation. These characteristics present many new challenges to previous intrusion detection solutions [42]. To deal with these issues, we design a DSGRU-based intrusion detection scheme, whose system framework is demonstrated in Figure 3. In our system framework, the main constituent parts can be roughly divided into four sections: data processing, model training, model testing, and result analysis.

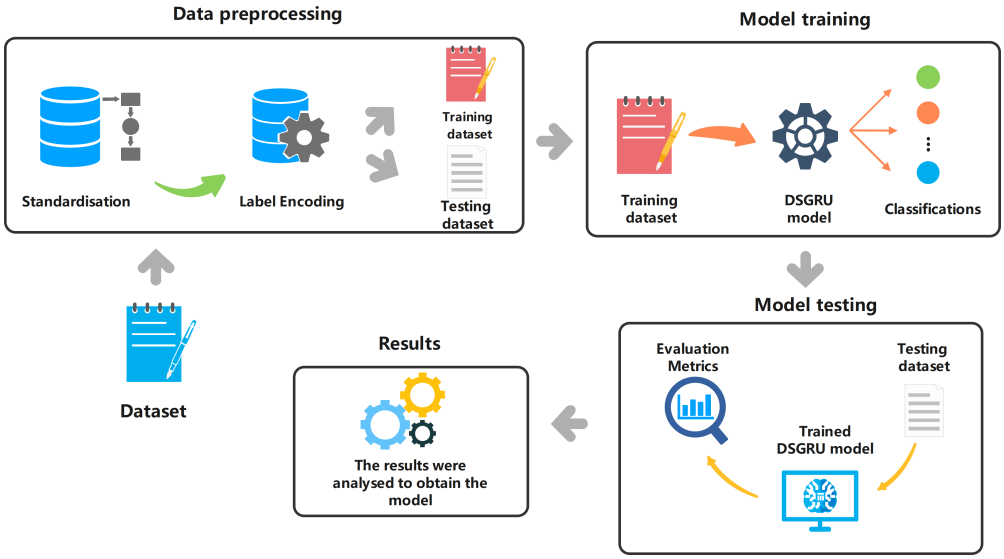

Figure 3. The system framework of our proposed scheme.

The data processing part primarily handles the original input data, i.e., dividing the original data into training and testing datasets, as well as achieving data redundancy elimination, digital encoding, feature extraction, and so on. The model training part focuses on training the DSGRU-based intrusion detection model using the training dataset, while the model testing part tests the trained DSGRU-based intrusion detection model using the testing dataset. That is, the model training part and the model testing part are designed to establish the DSGRU-based intrusion detection scheme, which achieves efficient and accurate intrusion detection in the IoT environment. The result analysis part enables the analysis of intrusion detection results, thereby demonstrating the efficiency, practicability, and accuracy of our proposed DSGRU-based intrusion detection scheme.

To demonstrate the DSGRU-based intrusion detection scheme implementation process more intuitively, the above steps can be summarized into an algorithmic pseudo-code, as shown in Algorithm 1.

Algorithm 1: DSGRU algorithm.

Input: x_{input}
Output: x_t

The dataset constitutes x_{input} data through feature standardization and feature numerical encoding
$x_{out} \leftarrow x_{input}$
for $d = 1$ *to* n **do**
    $x_t \leftarrow OGRU(x_{out})$
    $x_{out} \leftarrow DGRU(x_t)$
$lc \leftarrow x_{input} - x_{out}$
$lc_{out} \leftarrow OGRU(lc)$
return $x_t \leftarrow softMax(x_t + lc_{out})$

4.2. DSGRU-Based Intrusion Detection Scheme

In this subsection, we introduce the DSGRU-based intrusion detection scheme in detail. Note that compared with other neural network algorithms, DSGRU can greatly reduce feature loss through loss compensation, thereby improving robustness and generalization ability, and can better process the potential relation between long-term and short-term dependencies in temporally correlated data. Therefore, our presented DSGRU-based intrusion detection scheme can achieve better performance.

To demonstrate the proposed scheme more clearly, we utilize an algorithm to present the proposed scheme, as demonstrated in Algorithm 2.

Algorithm 2: Intrusion detection algorithm.

Input: Data Feature
Output: Classifications
Load Dataset
Data processing
for *d = 1 to n* **do**
    if *Feature = Nonnumerical* **then**
        Using numerical encoding
    else
        Feature standardization with $Z = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$
Start model training and testing
for *d=1 to n* **do**
    $DSGRU \leftarrow DSGRU_i(X)$;
    $result \leftarrow Compare(Y_{real}, Y_{pred})$
Test model on test sets
Calculate DSGRU model performance metrics

In Algorithm 2, we respectively utilize symbol x to represent the attribute features of the data; y to represent the category features of the data; use X to represent the attribute characteristics of the data in standardization. Meanwhile, we denote the minimum value by $X_{\min}$, denote the maximum value by $X_{\max}$, and denote the real situation value by Y_{real}. Moreover, Y_{pred} is utilized to represent the predicted situation value, N_x is used to represent the training epoch, and the symbol *resulting* is adopted to represent the evaluation results. The primary phases of Algorithm 2 can be summarized as follows.

(1) *Data pre-processing.* We process the original network traffic dataset. To be more specific, the original dataset will be transformed into numerical features and then normalized. Subsequently, we utilize the processed dataset to establish an intrusion detection network traffic dataset. The processed dataset will be divided into two parts: a training dataset and a testing dataset. The training dataset is utilized to train the DSGRU-based intrusion detection model and the testing dataset is used to test the intrusion detection performance.

(2) *Model training.* The training dataset is input into the DSGRU-based intrusion detection training model and executes n epochs of training. Subsequently, a trained DSGRU-based intrusion detection model is output.

(3) *Model testing.* The testing dataset is input into the trained DSGRU-based intrusion detection model and then output the intrusion detection results.

(4) *Result analysis.* According to the intrusion detection results output by the trained DSGRU-based intrusion detection model, some related metrics are calculated to evaluate the performance of the proposed scheme.

5. Computational Complexity

In the following part, we will analyze the theoretical computational complexity of the DSGRU-based intrusion detection scheme, SGRU-based intrusion detection scheme [40], GRU-based intrusion detection scheme [41], BiLSTM-based intrusion detection scheme [37], and DAE-BPNN-based intrusion detection scheme [39], as demonstrated in Table 1. For simplicity, symbol m is used to represent the dimensionality of the input data and symbol n is utilized to represent the dimensionality of the hidden layer.

Table 1. Theoretical computational complexity.

Methodology	Total Times of Calculations	Computational Complexity
DSGRU	$T[(2 \times d + 1) \times (3 \times n \times m + 3 \times n^2 + 3 \times n) + 2 \times n + 2 \times m]$	$O(n^2)$
SGRU [40]	$T[3 \times (3 \times n \times m + 3 \times n^2 + 3 \times n) + 2 \times n + 2 \times m]$	$O(n^2)$
GRU [41]	$T(3 \times n \times m + 3 \times n^2 + 3 \times n)$	$O(n^2)$
BiLSTM [37]	$T(2 \times (4 \times n \times m + 4 \times n^2 + 4 \times n))$	$O(n^2)$
DAE-BPNN [39]	$T(3 \times n \times m + 3 \times n^2))$	$O(n^2)$

To compute the time complexity of DSGRU, the time complexity of GRU is analyzed first. According to Formulas (1)–(4) in Section 2, GRU requires maintaining three sets of parameters: the input gate, the forgetting gate, and the candidate state. Therefore, the total computational time overhead of GRU is $T(3 \times n \times m + 3 \times n^2 + 3 \times n)$. That is, the computational complexity of GRU can be denoted as $O(n^2)$. LSTM then requires maintaining four sets of parameters: the input gate, the oblivion gate, the output gate, and the candidate state. Therefore, the total computational time overhead of BiLSTM is $T(2 \times (4 \times n \times m + 4 \times n^2 + 4 \times n))$. That is, the computational complexity of BiLSTM is denoted as $O(n^2)$. Meanwhile, the total computational time overhead of DSGRU is $T[(2 \times d + 1) \times (3 \times n \times m + 3 \times n^2 + 3 \times n) + 2 \times n + 2 \times m]$, where d is the number of layers of OGRU and DGRU. As a consequence, the computational complexity of DSGRU and GRU are the same and both are $O(n^2)$. Moreover, since both DAE and BPNN contain two-layer–fully connected layer structures in our reproduction experiments, the total computational time overhead is $T(3 \times n \times m + 3 \times n^2)$, and the computational complexity can be denoted as $O(n^2)$.

6. Experiments

In the following section, we will implement our proposed scheme and provide the performance evaluation. Specifically, we will provide a detailed description of the experimental environment, experimental dataset, and experimental evaluation metrics. Subsequently, we describe the experimental results, along with a comprehensive analysis.

6.1. Implementation

The proposed scheme will be implemented in the Python programming language using the PyTorch deep learning framework. The Adaptive Moment Estimation (Adam) algorithm is the optimizer used to update the model weights at a learning rate of 0.001. The number of neurons in the OGRU hidden layer is 20 and the number of neurons in the DGRU hidden layer unit is 43; meanwhile, the loss function used in the model is the categorical cross-entropy for multi-class classification. Moreover, the model batch size is set to 512 and the model training epoch is 250. The following two datasets (UNSW-NB15 dataset and NSL-KDD dataset) both match this parameter configuration. As shown in Figure 2, the model first uses an embedding layer to map the inputs to its representation. Then, it provides the embedding to the OGRU and DGRU layers with multilayer processing. The DSGRU output is then fed to the fully connected layer with Softmax as the activation function. Ideally, the fully connected layer learns and compiles the data extracted by the DSGRU layer to form the final output, which is classified by the output layer. Finally, we use loss compensation to reduce the loss of nonlinear transformation.

6.2. Experimental Environment

To simulate our proposed scheme, all the simulation experiments were performed on a desktop computer that was equipped with an Intel Core i5-8300H CPU running at 2.30 GHz, an NVIDIA GeForce GTX 1050 Ti graphics card, and 16 GB of main memory. Meanwhile, the desktop computer was also equipped with a Windows 11 operating system,

CUDA10.0 driver, cuDNN 7.4.2, and a Python 3.7 environment. Moreover, the deep learning framework utilized in the experiments was PyTorch 1.8.

6.3. Experimental Dataset

We utilized UNSW-NB15 and NSL-KDD as the experimental datasets.

6.3.1. UNSW-NB15 Dataset

The UNSW-NB15 dataset was created by the Australian Centre for Cyber Security (ACCS) lab in 2015 [43]. The UNSW-NB15 dataset comprises both real normal traffic data and malicious attack traffic data. Meanwhile, the UNSW-NB15 dataset encompasses normal traffic data and nine types of attack traffic data. The nine types of attack traffic data are backdoor, penetration analysis, denial of service (Dos) attack, vulnerability exploitation, fuzz test, generalized attack, stomp, shellcode, and worm. Generally, the UNSW-NB15 dataset consists of 2,540,044 data samples classified into 10 types, as follows:

- Normal: The normal network traffic data.
- Fuzzers: Attacks that attempt to paralyze a system with randomly generated data from the web.
- Analysis: Web attacks through port scanning, spamming, and file exfiltration.
- Backdoors: Hacking attacks that access computer data by bypassing system security mechanisms.
- DoS: Uninterrupted malicious attacks on the host computer intended to render the computer out of service.
- Exploits: Malicious attacks through the operating system or software security vulnerabilities.
- Generic: Group password hacking attacks.
- Reconnaissance: Attackers gather information related to their target to carry out the attack.
- Shellcode: Attackers use shell commands to control the target host attack methods.
- Worms: An attack method in which an attacker spreads to other computers through self-replication, rendering other computers unable to function properly.

To thoroughly evaluate the effectiveness of DSGRU in intrusion detection for the IoT environment, we carried out the experiments by using a randomized extraction method on a dataset that consisted of 600,000 samples from the UNSW-NB15 dataset. Meanwhile, to achieve proper evaluation, we split the dataset into a training dataset and a testing dataset, at a ratio of 5:1. Specifically, the first 500,000 data samples were utilized as the training dataset, while the remaining 100,000 data samples were designated as the testing dataset. By this rigorous approach, we can rigorously evaluate the overall performance of the DSGRU-based intrusion detection scheme for the IoT environment.

6.3.2. NSL-KDD Dataset

The NSL-KDD dataset is one of the benchmark datasets used for evaluating intrusion detection systems [44]. It is an enhanced form of the KDD Cup 99 dataset. There are five categories in the NSL-KDD dataset: Normal, Denial of Service (DoS) Attack, User to Root Attack (R2L), Remote to Local Attack (U2R), and a Probe attack. The data types are specifically described as follows:

- Normal: Normal network data.
- DoS: Uninterrupted malicious attacks on the host computer to render the computer out of service.
- R2L: Hacker attacks that access computer data by bypassing security system mechanisms.
- U2R: Attacks that attempt to paralyze the system through randomly generated data over the network.
- Probe: Web attacks through port scanning, spamming, and file exfiltration.

The NSL-KDD dataset contains KDDTrain^{+} as a training set for model learning, and KDDTesn^{+} and KDDTest^{-21} as testing sets for the performance evaluations of the trained models [45]. Among them, KDDTrain^{+} has 125,973 traffic samples, KDDTesn^{+} has 22,544 traffic samples, and KDDTest^{-21} has 11,850 samples. Moreover, to simulate the realism of network traffic, the testing dataset includes many attacks that do not appear in the training set. Therefore, except for the 22 attack types in the training set, there are 17 different attack types in the testing set. To better train the model, KDDTrain^{+} is mainly used as the training set and KDDTest^{+} as the testing set in the experiments.

6.4. Evaluation Metrics

To evaluate an intrusion detection scheme, the typical evaluation metrics include accuracy, false alarms, and missed alarms [46]. However, due to the serious imbalance of IoT network traffic data, normal network traffic data account for the vast majority, which would lead to the bias of traditional evaluation metrics. As a result, to guarantee the comprehensiveness of the evaluation, we will comprehensively adopt $Confusion\ Matrix$, $Accuracy$, $Precision$, $Recall$, and $F1\_score$ as evaluation metrics [47].

TruePositive (TP): True positive represents the number of samples that are truly attacks and that have been correctly detected as attacks.

FalsePositive (FP): False positive represents the number of attacks that are misclassified as normal samples.

TrueNegative (TN): True negative represents the number of normal samples that are incorrectly detected as attacks.

FalseNegative (FN): False negative represents the number of normal samples that are accurately detected as normal samples.

Accuracy: Accuracy represents the proportion of correctly detected samples to the total number of samples. The calculation formula of accuracy is as follows.

$$Accuracy = \frac{TN + TP}{TP + TN + FP + FN} \tag{13}$$

Precision: Precision indicates the proportion of samples detected by the model to be in the positive category that are actually in the positive category. It focuses on the accuracy of the model's predictions. The calculation formula of precision is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

Recall: Recall represents the proportion of correct detections that are positives in the model, as a proportion of the data that are actually positives. In the experiment, recall refers to the proportion of attacks that are correctly categorized, such as a proportion of the actual attack data, which can be calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$F1\_score$: $F1\_score$ is a comprehensive metric that balances precision and recall. $F1\_score$ is used to assess the model's performance by considering both $Precision$ and $Recall$ simultaneously. The $F1\_score$ is a value between 0 and 1; a higher value indicates a better model performance when $Precision$ and $Recall$ are close. The formula for the $F1\_score$ is as follows:

$$F1\_score = 2 \times \frac{precision \times recall}{recall + precision} \tag{16}$$

$Confusion\ Matrix$: The confusion matrix is a table used to evaluate the performance of a classification model. The rows and columns of the matrix, respectively, represent the predicted and actual values. Specifically, TP, FP, TN, and FN can be calculated based on the number of instances in each category of the confusion matrix, and then the evaluation

metrics, such as accuracy, precision, recall, and *F*1_*score* score can be calculated using Formulas (13)–(16).

6.5. Experimental Results

In this subsection, we simulate our proposed scheme and compare the performance with some previous intrusion detection solutions. Specifically, we will compare the performances among our proposed DSGRU-based intrusion detection scheme, SGRU-based intrusion detection scheme [40], GRU-based intrusion detection scheme [41], BiLSTM-based intrusion detection scheme [37], and DAE-BPNN-based intrusion detection scheme [39].

Confusion Matrix. We firstly test the fully trained model and then obtain the confusion matrix, as demonstrated in Figures 4 and 5.

In Figures 4 and 5, the diagonal of the confusion matrix represents the number of correct identifications for each category. For a deep learning model, the training process is a form of self-learning. In other words, deep learning models can learn a significant amount of knowledge from the training dataset and then present the data features. Then, in Figures 4 and 5, the larger the numbers in the diagonal, the more knowledge the model has learned. Because the dataset is extremely unbalanced, categories with more data can achieve higher accuracy, while those with less data tend to have lower accuracy.

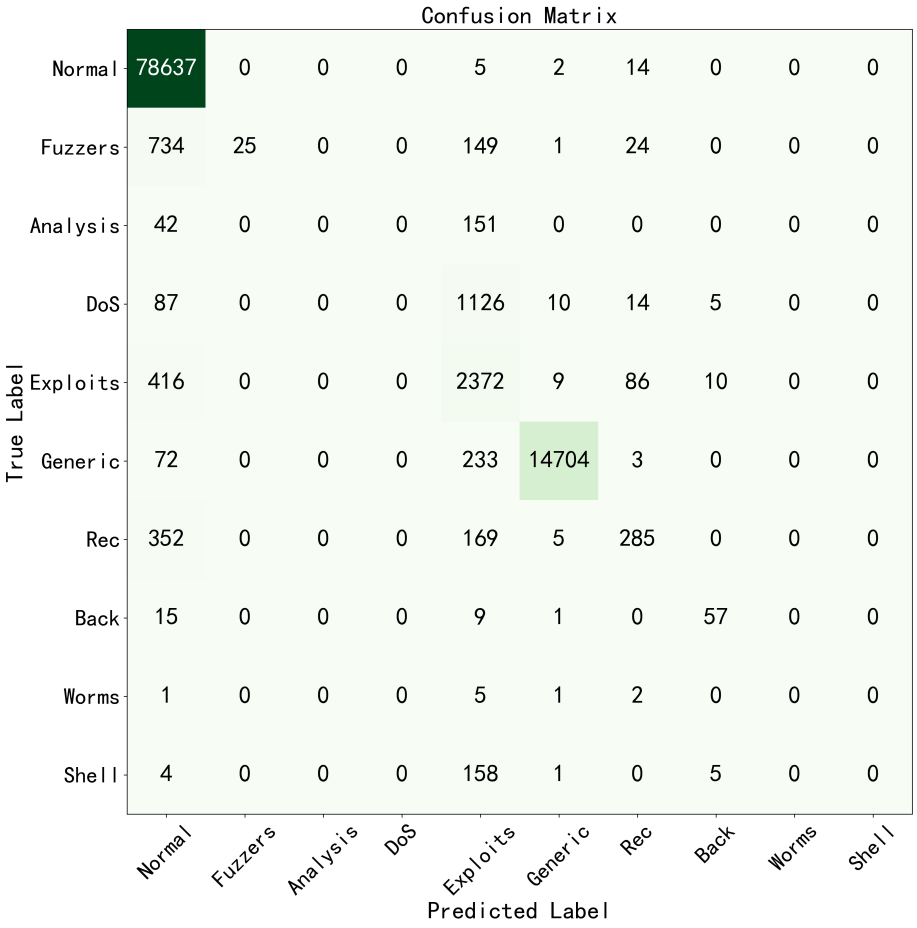

Figure 4. Confusion matrix of DSGRU based on the UNSW-NB15 dataset.

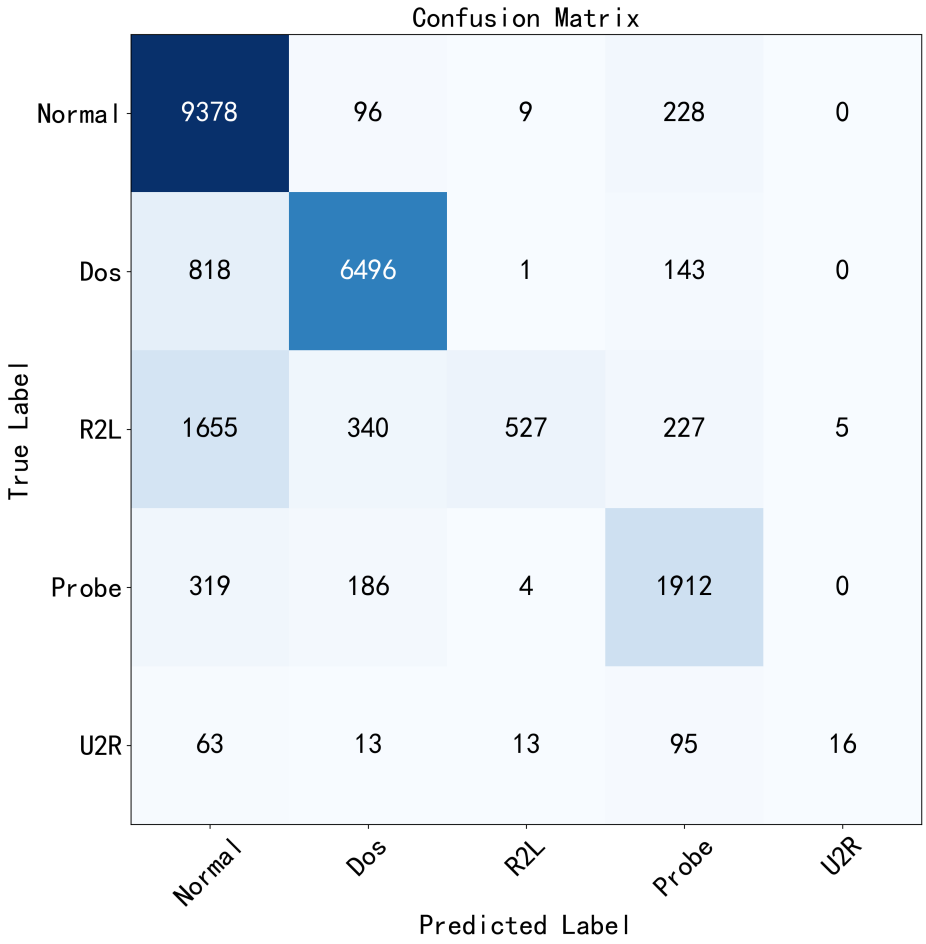

Figure 5. Confusion matrix of DSGRU based on the NSL-KDD dataset.

6.5.1. Model Fit Analysis

To evaluate the model fitting degree, we compare the loss values of the DSGRU-based intrusion detection scheme, SGRU-based intrusion detection scheme [40], GRU-based intrusion detection scheme [41], BiLSTM-based intrusion detection scheme [37], and DAE-BPNN-based intrusion detection scheme [39]. Subsequently, the comparison results are demonstrated in Table 2 and Figure 6.

Table 2. Comparison of loss values.

Dataset		DSGRU	SGRU	GRU	BiLSTM	DAE-BPNN
UNSW-NB15	Loss value	1.55^{-2}	1.62^{-2}	1.97^{-2}	1.95^{-2}	1.87^{-2}
NSL-KDD		1.76^{-3}	2.07^{-3}	1.82^{-3}	2.39^{-2}	3.39^{-3}

From Table 2, we can see that when the dataset is the UNSW-NB15 dataset, the final loss value of the GRU-based intrusion detection scheme [41] is the highest, followed by that of the BiLSTM-based intrusion detection scheme [37], and the loss value of our proposed DSGRU-based intrusion detection scheme is the lowest. However, when the dataset is the NSL-KDD dataset, the final loss value of the DAE-BPNN-based intrusion detection scheme [39] is the highest, followed by that of the BiLSTM-based intrusion detection scheme [37], and the loss value of our proposed DSGRU-based intrusion detection scheme is the lowest. This demonstrates that, compared with the other four previous schemes, our proposed DSGRU-based intrusion detection scheme can achieve a better degree of fit.

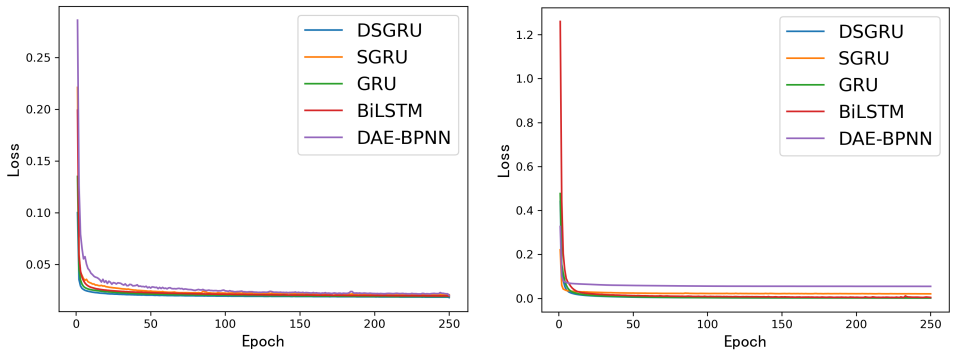

Figure 6. Comparison of loss values of different methods. (**a**) Loss for the UNSW-NB15 dataset. (**b**) Loss for the NSL-KDD dataset.

From Figure 6, we can obviously detect that our proposed DSGRU-based intrusion detection scheme has a better decline in loss value. In other words, compared with the other four previous solutions, our proposed scheme has a better loss value convergence rate and lower loss value. This is because—compared with the other four previous schemes—DSGRU can learn the temporal correlation of the dataset and achieve loss compensation in the model training process. Meanwhile, DSGRU can better consider the information to be transmitted in the future compared to the other four schemes.

6.5.2. Effectiveness Evaluation

To evaluate the effectiveness, we compare the five different schemes from the metrics of accuracy, precision, recall, and *F*1_*score* based on the UNSW-NB15 dataset and NSL-KDD dataset. Subsequently, the comparison results are demonstrated in Figure 7.

Figure 7a demonstrates the effectiveness of the comparison results based on the UNSW-NB15 dataset. We can obviously see that, compared with the other four previous intrusion detection schemes, our proposed scheme performs best in accuracy, precision, recall, and *F*1_*score*. For example, the accuracy of our proposed scheme reaches 96.08%, which is 1.34%, 1.86%, 1.74%, and 1.70% higher than those of the SGRU-based intrusion detection scheme [40], GRU-based intrusion detection scheme [41], BiLSTM-based intrusion detection scheme [37], and DAE-BPNN-based intrusion detection scheme [39], respectively. For another example, the precision of our DSGRU-based intrusion detection scheme reaches 95.52%. However, the precision of the SGRU-based intrusion detection scheme [40] is 92.89%, the precision of the GRU-based intrusion detection scheme [41] is 92.59%, the precision of the BiLSTM-based intrusion detection scheme [37] is 91.59%, and the precision of the DAE-BPNN-based intrusion detection scheme [39] is 90.89%. Meanwhile, the other two metrics (i.e., recall and *F*1_*score*) of our proposed scheme are both better than those of the other four solutions.

Meanwhile, to illustrate the good generalization ability and robustness of our proposed scheme, we also utilize the NSL-KDD dataset to verify the effectiveness; the comparison results are shown in Figure 7a. From Figure 7b, we can see that our proposed scheme is better than the other four solutions. For accuracy, our proposed scheme reaches 81.10%. However, the accuracies of the SGRU-based intrusion detection scheme [40], GRU-based intrusion detection scheme [41], BiLSTM-based intrusion detection scheme [37], and DAE-BPNN-based intrusion detection scheme [39] are 79.47%, 77.42%, 79.92%, and 78.24%, respectively. Similarly, the other metrics (i.e., precision, recall, and *F*1_*score*) of our proposed scheme all are better than those of the other four solutions.

From Figure 7, we see that our proposed scheme achieves the best performance among the five solutions, which means that DSGRU can more fully extract the related features

from the dataset. Meanwhile, DSGRU can better mine the intrinsic connection between the temporal correlation network traffic data.

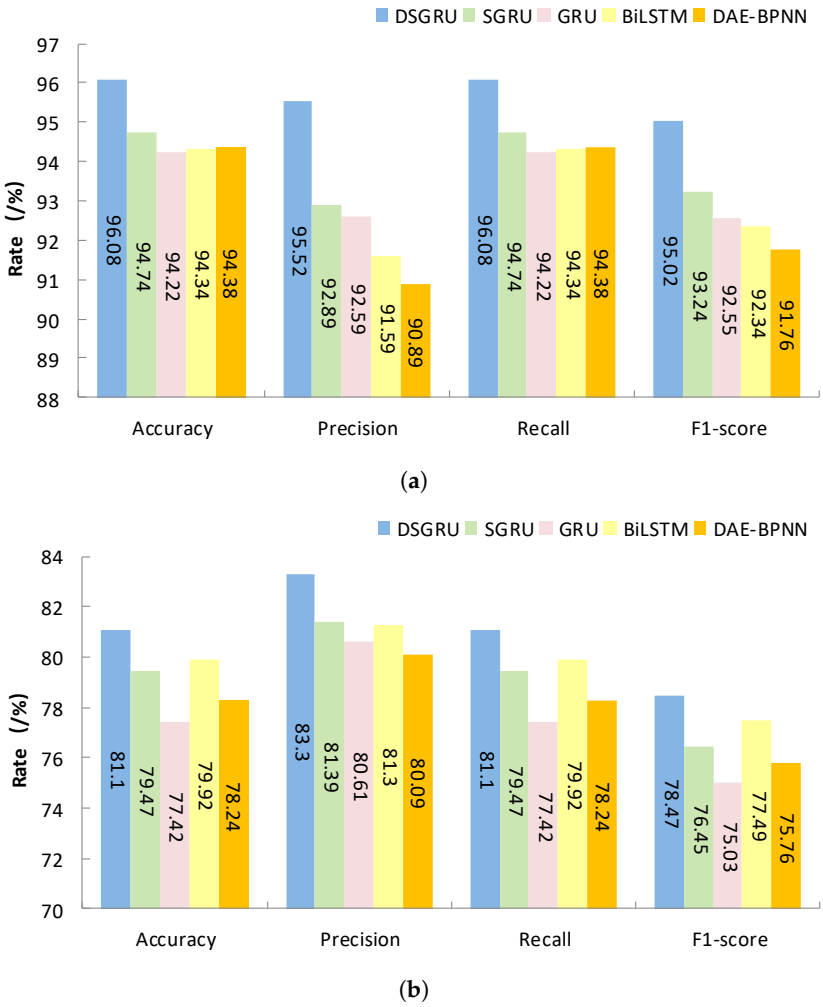

Figure 7. Comparison of the effectiveness of the five models. (**a**) Results for the UNSW-NB15 dataset. (**b**) Results for the NSL-KDD dataset.

6.5.3. Efficiency Evaluation

In this experiment, we, respectively, measure the total computational time overhead, the model training computational time overhead, and the model testing computational time overhead based on the theoretical computational complexity described in Table 1. Subsequently, the comparison results are demonstrated in Figures 8–10.

Efficiency evaluation based on the UNSW-NB15 dataset. We first measure the total time overhead and model training time overhead based on the UNSW-NB15 dataset, as shown in Figure 8. We can easily discover from Figure 8a that the BiLSTM-based intrusion detection scheme [37] requires the most total computational time overhead, followed by our proposed DSGRU-based intrusion detection scheme. The GRU-based intrusion detection scheme [41] requires the least total computational time overhead. This is because LSTM includes four gate structures, whereas GRU has only three gate structures and fewer parameters. Meanwhile, both DSGRU and SGRU comprise multiple GRUs, resulting in longer total computational time overhead compared to GRU, but still shorter than that of BiLSTM.

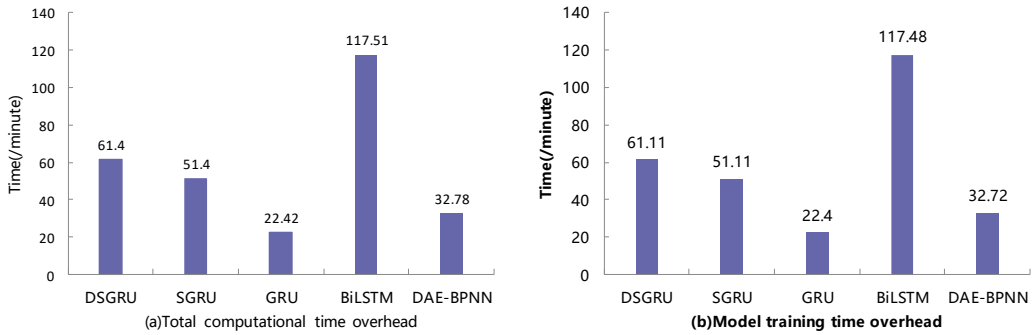

Figure 8. Total time overhead and model training time overhead based on the UNSW-NB15 dataset.

Figure 8b shows the model training time overhead, from which we can directly observe that the ranking of the model training time overhead for the five schemes is the same as that of the total time overhead. Meanwhile, from Figure 8, we see that model training constitutes the majority of the time overhead. For instance, the total computational time overhead of our proposed DSGRU-based intrusion detection scheme based on the UNSW-NB15 dataset is 61.40 min, while the model training requires 61.11 min. For another example, the BiLSTM model training requires 117.48 min, which occupies 99.97% of the total computational time overhead. In the model training process, feature extraction and learning will be completed. These two processes require some complex operations to help the model learn the features, hence they consume most of the time. However, it is important to note that model training is completed offline in advance, so it does not significantly affect efficiency.

To evaluate the real-time intrusion detection, we measure the model testing time overhead, as demonstrated in (Figure 9). From Figure 9, we can see that the BiLSTM-based intrusion detection scheme [37] requires the most model testing time overhead, followed by the DAE-BPNN-based intrusion detection scheme [39] and our proposed DSGRU-based intrusion detection scheme. The GRU-based intrusion detection scheme [41] requires the least model testing time overhead. However, the difference in model testing time overhead is small. For example, our proposed scheme requires 2.1 s to achieve model testing, while the model testing time overhead of the GRU-based intrusion detection scheme [41] is 1.3 s. In other words, the difference in model testing time overhead is so small that it is acceptable. Therefore, we can consider our proposed DSGRU-based intrusion detection scheme to be quite efficient.

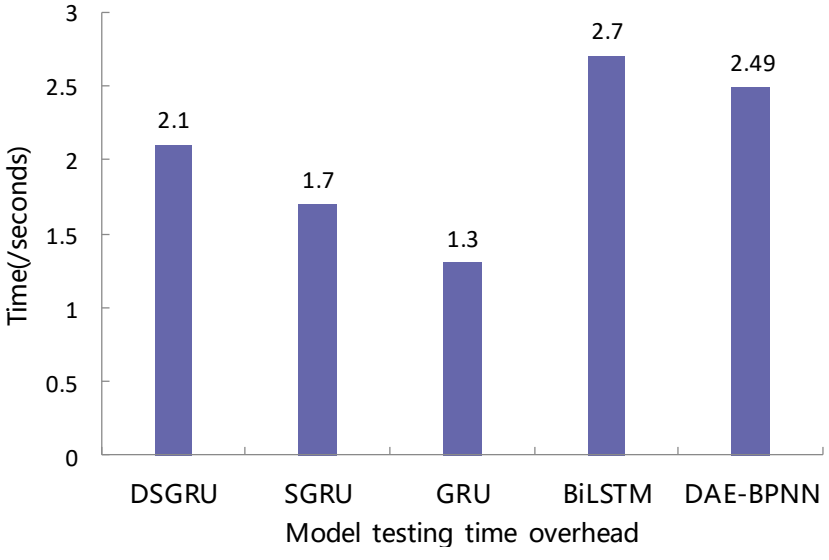

Figure 9. Model testing time overhead based on the UNSW-NB15 dataset.

Efficiency evaluation based on the NSL-KDD dataset. To further verify the efficiency of our proposed scheme, we also measure the time overhead based on the NSL-KDD dataset. Subsequently, the total time overhead and model training time overhead are demonstrated in Figure 10. We can clearly see that the time overhead for all five solutions is less when based on the NSL-KDD dataset compared to those based on the UNSW-NB15 dataset. This is because the number of data items in the NSL-KDD dataset is fewer than in the UNSW-NB15 dataset. Meanwhile, we see that the rankings of the total time overhead and the model training time overhead based on the NSL-KDD dataset are the same as those based on the UNSW-NB15 dataset. That is, the BiLSTM-based intrusion detection scheme [37] requires the most total computational time overhead and model training time overhead, followed by our proposed DSGRU-based intrusion detection scheme. The GRU-based intrusion detection scheme [41] requires the least total computational time overhead and model training computational time overhead.

Moreover, we measure the model testing time overhead based on the NSL-KDD dataset, as shown in Figure 11. From Figure 11, we can see that the DAE-BPNN-based intrusion detection scheme [39] requires the most model testing time overhead, followed by the BiLSTM-based intrusion detection scheme [37], and our proposed DSGRU-based intrusion detection scheme requires the least model testing time overhead. Specifically, in the model testing process, our proposed scheme merely requires 0.31 s, while the DAE-BPNN-based intrusion detection scheme [39] requires 1.08 s, and the BiLSTM-based intrusion detection scheme [37] requires 0.86 s. Meanwhile, the time overhead of our proposed scheme is 0.11 s less than that of the GRU-based intrusion detection scheme [41]. Therefore, we can see that our proposed DSGRU-based intrusion detection scheme is the most efficient in the model testing process based on the NSL-KDD dataset.

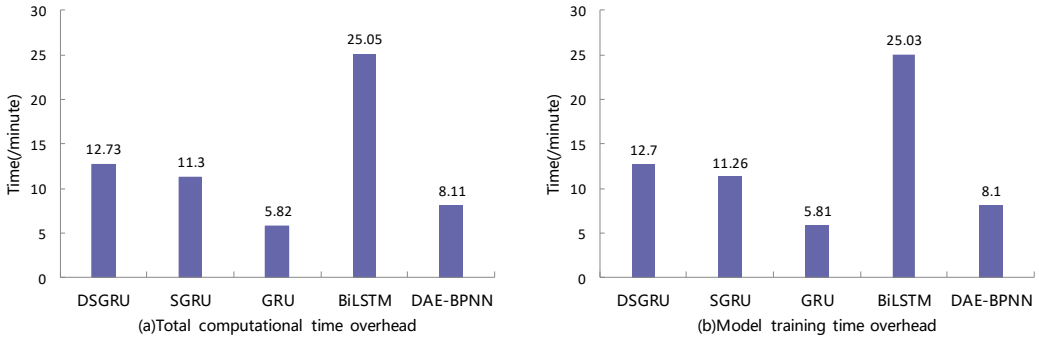

Figure 10. Total time overhead and model training time overhead based on the NSL-KDD dataset.

By analyzing and comparing the theoretical computational costs and computational complexity (which are described in Section 5), we see that although our proposed DSGRU-based intrusion detection scheme requires performing the most operations, the computational complexity of our scheme is the same as the other four previous schemes. Meanwhile, from the experimental results, we can see that our proposed DSGRU-based intrusion detection scheme requires a little more time overhead than the SGRU-based intrusion detection scheme [40] and GRU-based intrusion detection scheme [41]. However, the difference in time overhead is quite small. Moreover, our proposed DSGRU-based intrusion detection scheme can greatly improve the overall performance. Therefore, we can still consider our proposed scheme to be more attractive than the other four previous schemes

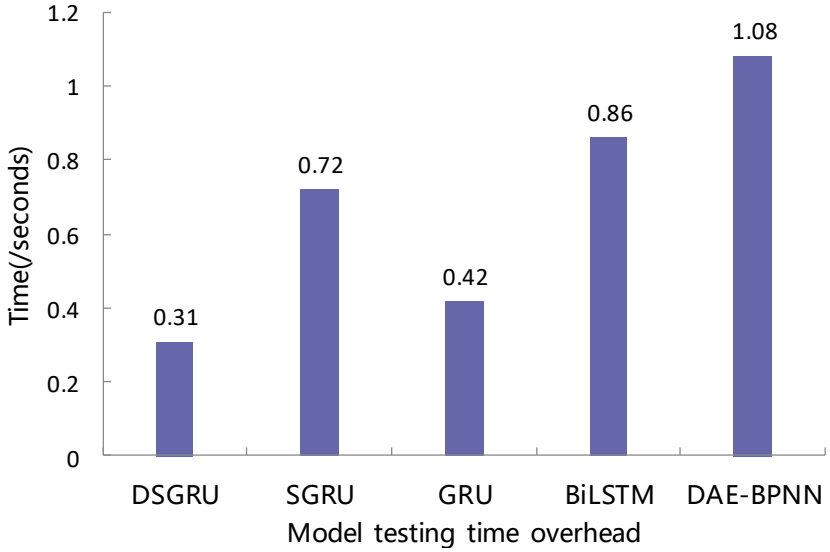

Figure 11. Model testing time overhead based on the NSL-KDD dataset.

7. Discussion

As is well known, there are many neural network models that can achieve intrusion detection. However, in this paper, DSGRU was designed and utilized to propose a novel intrusion detection scheme because DSGRU is perfectly suitable for the IoT environment, where the network traffic data is characterized by large-scale, heterogeneous multiple sources, high-dimensionality, and temporal correlation. In general, the main purpose of our proposed scheme is to attempt to address the following challenging problems.

How can the proposed scheme be suitable for the IoT environment compared to other comparable solutions? In the IoT environment, network traffic data are characterized by large-scale, heterogeneous multiple sources, high dimensionality, and temporal correlation. Traditional intrusion detection schemes require plenty of nonlinear transformations and complex matrix manipulations to process these network traffic data, thus reducing the efficiency and effectiveness. In our proposed scheme, we adopt DSGRU to establish the intrusion detection scheme, which can reduce the nonlinear transformations and matrix manipulations. Meanwhile, DSGRU can effectively find relations within the temporal correlation data. Therefore, by using the advantages of DSGRU, our proposed scheme can improve the overall performance in processing large-scale, heterogeneous multiple sources, high-dimensionality, and temporal correlation network traffic data.

How can the proposed scheme effectively reduce feature loss compared to the other neural network model? In our proposed scheme, we adopt DSGRU as the building block to establish the intrusion detection scheme. Generally, DSGRU consists of ORU, DGRU, and the Softmax activation function. On the one hand, DSGRU can learn more features by improving the number of layers for OGRU and DGRU. On the other hand, in the model training process, DSGRU will execute the loss compensation operation. That is, DSGRU will obtain the loss data and add them to x_t. As a consequence, our proposed scheme can effectively reduce feature loss during the model training process.

How can the proposed scheme greatly improve effectiveness compared to other comparable solutions? Compared with some previous solutions, our proposed scheme can greatly improve the effectiveness of intrusion detection due to the following factors. Firstly, DSGRU can perform loss compensation to reduce feature loss, thus improving the generalization ability of the intrusion detection model. Secondly, by increasing the layers of OGRU and DGRU, DSGRU can learn more feature data from the original dataset. Lastly, DSGRU can effectively reduce the operations of nonlinear transformations, thus reducing feature loss. As a consequence, our proposed scheme can greatly improve effectiveness compared to other comparable solutions.

8. Conclusions and Future Work

In this section, we will first conclude this paper. Subsequently, we will describe the research points that we will study in the future.

8.1. Conclusions

In an IoT environment, the network traffic data are characterized by large-scale, heterogeneous multiple sources, high-dimensionality, and temporal correlation, which could seriously affect the efficiency and accuracy of deep-learning-based intrusion detection schemes. To solve the above challenge, we designed a DSGRU-based intrusion detection scheme for the IoT environment. Specifically, we first designed a new neural network model called DSGRU. Subsequently, we utilized DSGRU to propose a novel intrusion detection scheme for the IoT environment. By using the attractive advantages of DSGRU, our proposed intrusion detection scheme could greatly improve the overall performance. Finally, we implemented our proposed DSGRU-based intrusion detection scheme and provided accurate experimental results based on the UNSW-NB15 dataset and NSL-KDD dataset. Meanwhile, the experimental results show that our proposed DSGRU-based intrusion detection scheme can achieve significant improvements in both effectiveness and efficiency when compared to some previous intrusion detection solutions. Specifically, when the datasets are UNSW-NB15 and NSL-KDD, the accuracies are 96.06% and 81.10%, the precisions are 95.52% and 83.34%, and the model training times are 61 min and 12 min, respectively.

8.2. Future Work

In this paper, we mainly studied the design of the intrusion detection scheme for the IoT environment and proposed a DSGRU-based intrusion detection scheme. Our proposed scheme can improve intrusion detection efficiency and accuracy. However, in practical applications, system administrators not only wish to comprehend the current security situation through intrusion detection but also seek to know the future security situation through intrusion forecasting. Therefore, we will study the solution for intrusion forecasting.

Author Contributions: Conceptualization, Y.L. (Yueling Liu) and C.Y.; methodology, Y.L. (Yueling Liu) and C.Y.; software, C.Y. and Y.L. (Yingcong Lan); validation, Y.L. (Yueling Liu), Y.L. (Yingcong Lan) and Y.D.; investigation, C.L.; resources, Y.D.; data curation, Y.L. (Yingcong Lan); writing—original draft preparation, Y.L. (Yueling Liu); writing—review and editing, Y.L. (Yueling Liu) and C.Y.; visualization, Y.L. (Yueling Liu); supervision, Y.D. and C.L.; project administration, C.L.; funding acquisition, C.Y. and Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Science and Technology Project of Guangxi (grant no. AA22068067), the Central Guidance on Local Science and Technology Development Fund of Guangxi Province (grant no. ZY23055008), the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (grant no. 2022B1212010005), the Guangxi Key Laboratory of Trusted Software (grant no. KX202329), and the National Natural Science Foundation of China (grant no. 62172119).

Data Availability Statement: The UNSW-NB15 dataset can be downloaded from https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 18 August 2023) and the NSL-KDD dataset can be downloaded from https://www.unb.ca/cic/datasets/nsl.html (accessed on 29 August 2023).

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

References

1. Xiong, J.; Ma, R.; Chen, L.; Tian, Y.; Li, Q.; Liu, X.; Yao, Z. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4231–4241. [CrossRef]
2. Bag, S.; Gupta, S.; Kumar, S. Industry 4.0 adoption and 10R advance manufacturing capabilities for sustainable development. *Int. J. Prod. Econ.* **2021**, *231*, 107844. [CrossRef]

3. Kou, L.; Wu, J.; Zhang, F.; Ji, P.; Ke, W.; Wan, J.; Liu, H.; Li, Y.; Yuan, Q. Image encryption for Offshore wind power based on 2D-LCLM and Zhou Yi Eight Trigrams. *Int. J. Bio-Inspired Comput.* **2023**, *22*, 53–64. [CrossRef]
4. Roman, R.; Zhou, J.; Lopez, J. On the features and challenges of security and privacy in distributed internet of things. *Comput. Netw.* **2013**, *57*, 2266–2279. [CrossRef]
5. Togay, C.; Kasif, A.; Catal, C.; Tekinerdogan, B. A firewall policy anomaly detection framework for reliable network security. *IEEE Trans. Reliab.*, **2022**, *71*, 339–347. [CrossRef]
6. Ren, H.; Xu, G.; Qi, H.; Zhang, T. PriFR: Privacy-preserving Large-scale File Retrieval System via Blockchain for Encrypted Cloud Data. In Proceedings of the 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), New York, NY, USA, 6–8 May 2023; IEEE: Piscataway, NJ, USA, 2023.
7. Ning, H.; Zhen, Z.; Shi, F.; Daneshm, M. A survey of identity modeling and identity addressing in Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 4697–4710. [CrossRef]
8. Taparia, A.; Banu, P.K.N. A survey of blockchain: Concepts, applications and challenges. *Int. J. Comput. Sci. Math.* **2023**, *17*, 152–165. [CrossRef]
9. Muhammad, K.; Hamza, R.; Ahmad, J.; Lloret, J.; Wang, H.; Baik, S.W. Secure surveillance framework for IoT systems using probabilistic image encryption. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3679–3689. [CrossRef]
10. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. [CrossRef]
11. Ren, H.; Li, H.; Liu, D.; Xu, G.; Cheng, N.; Shen, X. Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1052–1064. [CrossRef]
12. Verma, S.; Kawamoto, Y.; Kato, N. A network-aware Internet-wide scan for security maximization of IPV6-enabled WLAN IoT devices. *IEEE Internet Things J.* **2020**, *8*, 8411–8422. [CrossRef]
13. Arisdakessian, S.; Wahab, O.A.; Mourad, A.; Otrok, H.; Guizani, M. A survey on IoT intru-sion detection: Federated learning, game theory, social psychology, and explainable AI as future directions. *IEEE Internet Things J.* **2022**, *10*, 4059–4092. [CrossRef]
14. Zhang, J.; Wang, Y.; Li, S.; Shi, S. An architecture for IoT-enabled smart transportation security system: A geospatial approach. *IEEE Internet Things J.* **2020**, *8*, 6205–6213. [CrossRef]
15. Zhang, J.; Wang, Y.; Li, S.; Shi, S. POTA: Privacy-Preserving Online Multi-Task Assignment with Path Planning. *IEEE Trans. Mob. Comput.* **2023**, 1–13. . [CrossRef]
16. Hu, C.; Zhang, C.; Lei, D.; Wu, T.; Liu, X.; Zhu, L. Achieving Privacy-Preserving and Verifiable Support Vector Machine Training in the Cloud. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3476–3491. [CrossRef]
17. Zhang, C.; Hu, C.; Wu, T.; Zhu, L.; Liu, X. Achieving efficient and privacy-preserving neural network training and prediction in cloud environments. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 4245–4257. [CrossRef]
18. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
19. Kuang, F.; Xu, W.; Zhang, S. A novel hybrid KPCA and SVM with GA model for intrusion detection. *Appl. Soft Comput.* **2014**, *18*, 178–184. [CrossRef]
20. Mohammadi, M.; Rashid, T.A.; Karim, S.H.T.; Aldalwie, A.H.M.; Tho, Q.T.; Bidaki, M.; Rahmani, A.M.; Hosseinzadeh, M. A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. *J. Netw. Comput. Appl.* **2021**, *178*, 102983. [CrossRef]
21. Kuang, F.; Zhang, S.; Jin, Z.; Xu, W. A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. *Soft Comput.* **2015**, *19*, 1187–1199. [CrossRef]
22. Gu, J.; Lu, S. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Comput. Secur.* **2021**, *103*, 102158. [CrossRef]
23. Hussain, J.; Lalmuanawma, S.; Chhakchhuak, L. A two-stagehybrid classification technique for network intrusion detection system. *Int. J. Comput. Intell. Syst.* **2016**, *9*, 863–875. [CrossRef]
24. Lv, L.; Wang, W.; Zhang, Z.; Liu, X. A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine. *Knowl.-Based Syst.* **2020**, *195*, 105648. [CrossRef]
25. Saleh, A.I.; Talaat, F.M.; Labib, L.M. A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers. *Artif. Intell. Rev.* **2019**, *51*, 403–443. [CrossRef]
26. Zhang, C.; Costa-Pérez, X.P.; Patras, P. Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms. *IEEE/ACM Trans. Netw.* **2022**, *30*, 1294–1311. [CrossRef]
27. Xie, G.; Yang, L.T.; Yang, Y.; Luo, H.; Li, R.; Alazab, M. Threat analysis for automotive CAN networks: A GAN model-based intrusion detection technique. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 4467–4477. [CrossRef]
28. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A deep learning approach to network Intrusion detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
29. Otoum, S.; Kantarci, B.; Mouftah, H.T. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Netw. Lett.* **2019**, *1*, 68–71. [CrossRef]
30. Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Comput. Secur.* **2020**, *92*, 101752. [CrossRef]

31. Li, B.; Wu, Y.; Song, J.; Lu, R.; Li, T.; Zhao, L. DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5615–5624. [CrossRef]
32. Almutlaq, S.; Derhab, A.; Hassan, M.M.; Kaur, K. Two-stage intrusion detection system in intelligent transportation systems using rule extraction methods from deep neural networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, 1–15. [CrossRef]
33. Keshk, M.; Koroniotis, N.; Pham, N.; Moustafa, N.; Turnbull, B.; Zomaya, A.Y. An explainable deep learning-enabled intrusion detection framework in IoT networks. *Inf. Sci.* **2023**, *639*, 119000. [CrossRef]
34. Wu, Y.; Lee, W.W.; Gong, X.; Wang, H. A hybrid intrusion detection model combining SAE with kernel approximation in Internet of Things. *Sensors* **2020**, *20*, 5710. [CrossRef] [PubMed]
35. Lee, J.H.; Park, K.H. AE-CGAN model based high performance network intrusion detection system. *Appl. Sci.* **2019**, *9*, 4221. [CrossRef]
36. Lo W.; Alqahtani, H.; Thakur, K.; Almadhor, A.; Chander, S.; Kumar, G. A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. *Veh. Commun.* **2022**, *35*, 100471.
37. Imrana, Y.; Xiang, Y.; Ali, L.; Abdul-Rauf, Z. A bidirectional LSTM deep learning approach for intrusion detection. *Expert Syst. Appl.* **2021**, *185*, 115524. [CrossRef]
38. Aljrees, T.; Kumar, A.; Singh, K.U.; Singh, T. Enhancing IoT Security through a Green and Sustainable Federated Learning Platform: Leveraging Efficient Encryption and the Quondam Signature Algorithm. *Sensors* **2023**, *23*, 8090. [CrossRef]
39. Sood, K.; Nosouhi, M.R.; Nguyen, D.D.N.; Jiang, F.; Chowdhury, M.; Doss, R. Intrusion detection scheme with dimensionality reduction in next generation networks. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 965–979. [CrossRef]
40. Liu, Z.Y.; Yang, C.S.; Xiao, J.; Song, B.W.; Shi, K.X. A Novel Intrusion Detection Method Based on Supplement Gate Recurrent Unit for IoT. *Wirel. Communi-Cations Mob. Comput.* **2022**, *2022*, 3678493. [CrossRef]
41. Ansari, M.S.; Bartoš, V.; Lee, B. GRU-based deep learning approach for network intrusion alert prediction. *Future Gener. Comput. Syst.* **2022**, *128*, 235–247. [CrossRef]
42. Zavrak, S.; İskefiyeli, M. Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access* **2020**, *8*, 108346–108358. [CrossRef]
43. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference, Cracow, Poland, 18–19 May 2015.
44. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: Piscataway, NJ, USA, 2009.
45. Lee, W.; Stolfo, S.J. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur. (TiSSEC)* **2000**, *3*, 227–261. [CrossRef]
46. Peres, R.S.; Barata, J.; Leitao, P.; Garcia, G. Multistage quality control using machine learning in the automotive industry. *IEEE Access* **2019**, *7*, 79908–79916. [CrossRef]
47. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [CrossRef] [PubMed]