electronics

MDPI

Article

Intelligent Analysis and Prediction of Computer Network Security Logs Based on Deep Learning

Zhiwei Liu [1,2], Xiaoyu Li [1] and Dejun Mu [1,3,*]

[1] Network Security College, Northwestern Polytechnical University, Xi'an 710072, China; liuzhw@nwpu.edu.cn (Z.L.); lixiaoyu@nwpu.edu.cn (X.L.)
[2] Office of Information Construction and Management, Northwestern Polytechnical University, Xi'an 710072, China
[3] Shenzhen Research Institute, Northwestern Polytechnical University, Shenzhen 518057, China
* Correspondence: kevin@nwpu.edu.cn

Abstract: Since the beginning of the 21st century, the development of computer networks has been advancing rapidly, and the world has gradually entered a new era of digital connectivity. While enjoying the convenience brought by digitization, people are also facing increasingly serious threats from network security (NS) issues. Due to the significant shortcomings in accuracy and efficiency of traditional Long Short-Term Memory (LSTM) neural networks (NN), different scholars have conducted research on computer NS situation prediction methods to address the aforementioned issues of traditional LSTM based NS situation prediction algorithms. Although these algorithms can improve the accuracy of NS situation prediction to a certain extent, there are still some limitations, such as low computational efficiency, low accuracy, and high model complexity. To address these issues, new methods and techniques have been proposed, such as using NN and machine learning techniques to improve the accuracy and efficiency of prediction models. This article referred to the Bidirectional Gated Recurrent Unit (BiGRU) improved by Gated Recurrent Unit (GRU), and introduced a multi model NS situation prediction algorithm with attention mechanism. In addition, the improved Particle Swarm Optimization (PSO) algorithm can be utilized to optimize hyperparameters and improve the training efficiency of the GRU NN. The experimental results on the UNSW-NB15 dataset show that the algorithm had an average absolute error of 0.0843 in terms of NS prediction accuracy. The RMSE was 0.0932, which was lower than traditional prediction algorithms LSTM and GRU, and significantly improved prediction accuracy.

Keywords: intelligent analysis and prediction; computer network security log; deep learning; long short-term memory; Gated Recurrent Unit; Particle Swarm Optimization

Citation: Liu, Z.; Li, X.; Mu, D. Intelligent Analysis and Prediction of Computer Network Security Logs Based on Deep Learning. *Electronics* **2024**, *13*, 4556. https://doi.org/10.3390/electronics13224556

Academic Editor: Hung-Yu Chien

Received: 22 October 2024
Revised: 15 November 2024
Accepted: 16 November 2024
Published: 20 November 2024

1. Introduction

Since the beginning of the 21st century, the field of computer network has developed rapidly. The Internet has been more widely used, making people's work and life more efficient and convenient. At the same time, NS issues have become more widespread, and the privacy information stored in various websites and apps is at risk of being leaked [1–3]. In order to address the increasingly serious threat of NS issues, NS situational awareness technology [4–6] has attracted the attention of almost all relevant researchers since its inception. NS situation prediction refers to the analysis of network historical data in NS logs to predict the security of the network in advance for a period of time in the future. At the same time, it can provide warnings for potential threats and attacks on the future network, help relevant technical personnel to allocate network resources reasonably in advance, and take corresponding countermeasures against predicted threats [7].

NN models have become one of the mainstream directions in NS situation prediction research due to their strong learning ability and high accuracy. The application of deep learning (DL) NN models to NS situation prediction [8–10] has become a trend. NN

models can introduce nonlinear mappings and adjust the weights of mapping relationships through nonlinear activation functions. NN can utilize nonlinear activation functions to establish a nonlinear mapping, adjust the weights of mapping relationships, learn complex mappings between massive inputs and outputs, and achieve accurate prediction of NS situations [11,12]. Situation awareness was first proposed by American researcher Endsley, and this concept was first applied in the field of the Air Force, which means "predicting the future evolution trend of something through the analysis of historical events in a specific environment" [13]. Kasongo S M and other researchers used a DL-based feature extraction algorithm for NS situation prediction [14]. Zhang H and other scholars have proposed methods that combine business flow computation with frequent patterns, as well as support vector machine classification algorithms based on deep belief networks. The use of sliding window (SW) technology to achieve online monitoring is a commonly used technical means [15]. HIDS (Hybrid Intrusion Detection System) is a hybrid IDS (Intrusion Detection System) in the United States. He used a class of support vector machines to establish a C5 decision tree classifier to improve the false alarm rate of NS situations [16]. In addition, Gu C proposed the Random Forest (RF) algorithm to alleviate traffic imbalance [17]. Han Xiaolu et al. proposed a new NS situation prediction model with abnormal inputs based on intuitionistic fuzzy sets. On this basis, he structured the feature elements of NS big data and obtained an intuitive fuzzy set [18]. Boukhalfa A proposed a new Network Intrusion Detection System (NIDS) [19], which is an algorithm based on LSTM [20] to identify threats and prevent new attacks similar to existing ones [21]. Although it greatly improves the accuracy of algorithm detection, it greatly increases model complexity and reduces efficiency. The GRU [22] multi model prediction algorithm improved by Assis MVO on the LSTM algorithm has improved efficiency but decreased accuracy compared to the LSTM algorithm [23]. The above algorithms can only focus on one aspect of accuracy and efficiency, making it difficult to balance them.

This article draws on an improved gated loop unit based on attention mechanism, and constructs a multi model NS situation prediction algorithm by introducing attention mechanism into the improved gated loop unit. In order to accelerate the learning rate of GRU NN, this paper refers to an improved PSO [24,25], achieving the goal of balancing detection accuracy and detection efficiency.

2. Related Works

Network security log analysis plays an important role in protecting computer networks, and some scholars have conducted research on the issue of network security log analysis. Xiong W et al. proposed a matrix-based threat modeling language for analyzing network security logs in enterprise security. This language uses the Meta Attack Language framework to describe system assets, attack steps, defenses, and asset associations, simulating responses to attacker techniques by simulating attacks on model instances, study security settings and architecture changes to enhance system security. The research was validated through multiple unit and integrated tests, and the modeling and simulation processes of two real network attacks were presented in the paper [26]. Georgiadou A et al. proposed a network security culture framework for analyzing network security logs. The study comprehensively reviews commonly used security frameworks, identifies core human security related elements, and constructs a domain independent security model. It also provides a detailed description of each component of the model and quantifies it to form a feasible evaluation method. This model has strong adaptability and provides in-depth insights for applications in security-critical areas, emphasizing the importance of human factors in network security [27]. Rajadurai H and Gandhi U D proposed a stacked ensemble learning model for wireless network security log analysis. In view of the fact that traditional single algorithms face statistical and computational problems when dealing with large amounts of internet data, they combine multiple machine learning algorithms to improve the attack detection effect. The experimental results show that stacked ensemble

learning outperforms other methods in attack classification and significantly improves accuracy, providing an effective solution for network security detection [28].

Some scholars have also conducted research on deep learning technology. Ferrari Dacrema M and other scholars have proposed a deep learning-based method for generating a ranking item list in teaching recommendation systems. During the process, an analysis was conducted on collaborative filtering methods, and a baseline was selected for comparison. A matrix factorization model was used for calculation. The experimental results show that the proposed method has good recommendation quality [29]. Cong et al. proposed a module-based convolutional neural network architecture classification method. By analyzing the advantages and disadvantages of various neural network architectures and comparing their performance, and based on the mathematical principles of optimization algorithms, a comprehensive classification of network compression and acceleration network architecture optimization algorithms is carried out. The experimental results show that the proposed method can effectively identify neural network architectures in specific practical applications, and quickly find neural network algorithms that fit the problem [30]. Nguyen et al. proposed a deep learning method based on sound analysis for machine fault detection. By preprocessing and extracting features from the sound signals generated by the machine under different operating conditions, a convolutional neural network is used to automatically learn the required features for classification. The results show that the model exhibits high accuracy in fault detection of known and unknown machines, proving its good performance in machine fault detection [31]. Stupariu M S and other scholars have proposed a deep learning based method for addressing landscape design issues. During the process, quantitative statistics combined with cluster analysis are used to retrieve landscape design keywords, and nonmetric multidimensional scales of different design attributes are analyzed to describe the relationship between different design contents. The experimental results show that the proposed method has good design efficiency [32]. The comparison between research methods and recent studies is shown in Table 1.

Table 1. Comparison of Research Methods and Recent Studies.

Reference	Strengths	Weaknesses	Advantages of This Research Method
Xiong et al. [26]	Provides an effective model for attack simulation and response.	Complex model that is difficult to apply in practice.	This research achieves flexible prediction through GRU and attention mechanisms.
Georgiadou et al. [27]	Establishes a highly adaptable human factors model.	Too reliant on security culture, lacking in technical implementation.	This research emphasizes the combination of technology and human factors.
Rajadurai & Gandhi [28]	Ensemble learning enhances detection accuracy.	Complex model with high computational costs and limited real-time performance.	This research reduces model complexity and improves efficiency.
Ferrari Dacrema et al. [29]	Uses deep learning to generate high-quality recommendations.	Limited applicability, not related to network security.	This research focuses on network security issues.
Cong et al. [30]	Evaluates the best CNN architectures for optimization.	Not strongly related to network security.	This research improves prediction accuracy in network security.
Nguyen et al. [31]	Achieves high accuracy in fault detection through sound analysis.	Application limited to mechanical faults.	This research's GRU model is applicable to network security scenarios.
Stupariu et al. [32]	Enhances efficiency in landscape design.	Not applicable to the field of network security.	This research focuses on predictive analysis and response in network security contexts.

3. Research Motivation and Contribution

In the information age, network security threats are constantly evolving, and traditional security defense measures are no longer effective in dealing with complex and dynamic attack patterns. Therefore, research needs to shift towards more intelligent and adaptive ways to predict and respond to potential network attacks in real time. This study

aims to improve the detection and prediction capabilities of network security events by combining advanced deep learning methods such as GRU and attention mechanisms, thereby enhancing the overall efficiency of network security defense.

The contribution of the research lies in the development of a network security prediction model based on GRU and attention mechanism, which can more accurately identify potential threats in dynamic environments. By introducing deep learning techniques, the prediction accuracy of network attacks has been significantly improved, with better performance compared to methods in existing literature. The research method is not only applicable to various types of network attacks, but also demonstrates high flexibility and adaptability in real-time detection and response. The study integrated technology and human factors in model design, emphasizing the importance of deep integration of safety culture and technology, making the model more practical.

4. GRU Improved Algorithm

4.1. Attention-CNN-BiGRU Improved Algorithm

Attention-CNN-GRU (Attention-Convolutional Neural Network Bidirectional Gated Recurrent Unit Neural Network) combines GRU and attention mechanism (AM). Its design purpose is to introduce AM to enable the model to dynamically focus on information at different positions in the input sequence, in order to improve the modeling of the sequence. Attention-CNN-GRU introduces AM on the basis of having components such as reset gates, update gates, and candidate hidden states. The AM includes calculating attention weights and weighted summation. By utilizing the hidden state at the current moment and information from various positions in the input sequence, attention weights are calculated. Usually, dot product, additive, or other methods are used to calculate [33]. The calculated attention weights can be used to weight and sum the information of each position in the input sequence, forming a weighted context vector. Attention CNN GRU combines the calculated weighted context vector with the output of GRU to generate the final output flow as shown in Figure 1.

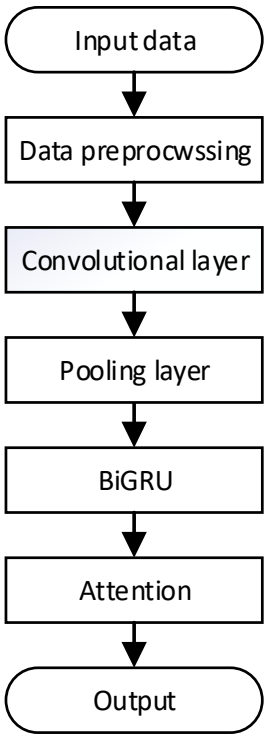

Figure 1. Attention CNN GRU algorithm flowchart.

The composition structure of the improved algorithm includes the input layer, GRU encoder, attention layer, and prediction network. The specific structure is as follows:

(1) Input layer

This study used the CICIDS2017 dataset in the experiment. Firstly, the selected dataset can be preprocessed and dimensionally adjusted to meet the requirements of network training. Next, the model can be trained by starting the input layer.

(2) Encoding layer

GRU, as the NN processing unit of the encoder, is an important module for sequence data processing. It obtains data from the input layer and provides the output hidden state values to the attention layer through a series of computational operations, thereby supporting further processing of the encoded data.

GRU is a variant of Recurrent Neural Network (RNN) specifically designed to solve the long sequence problem in traditional RNNs [34]. Similarly, the GRU algorithm also has gating structures, forgetting mechanisms, and memory cells, but GRU effectively captures and maintains information in the sequence by introducing update and reset gates. The structure of the GRU algorithm mainly includes reset gates, update gates, update hidden states, and candidate hidden states. The structure of the GRU model is depicted in Figure 2.

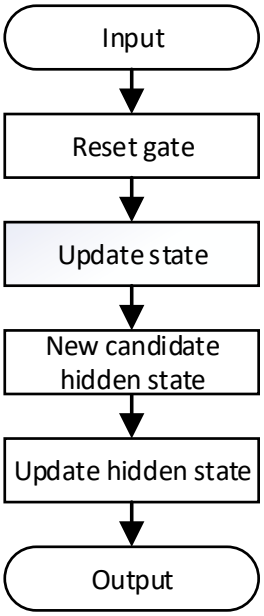

Figure 2. GRU model flowchart.

It generates an output between 0 and 1 by using the sigmoid function, as shown in Equation (1):

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (1)$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (2)$$

Among them, z_t is the output of the update gate, and W_z and b_z are the weights and biases of the update gate. The candidate hidden state $\overline{h}_t$ generates outputs between -1 and 1 by using the tanh function.

$$\overline{h}_t = tanh(W_h \cdot [h_{t-1}, x_t] + b_h) \quad (3)$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \overline{h}_t \quad (4)$$

The structure of GRU is relatively simplified, but it can still effectively capture relationships in sequences, avoiding explicit long-term memory units in LSTM. GRU is easier

to train in certain situations with fewer parameters and easier to deploy in resource limited environments.

In NN models, encoders play the role of transforming input data into higher-level representations. It receives data from the input layer and encodes the sequence data through the calculation of the GRU unit. The gating mechanism inside the GRU unit can determine which information should be remembered and which information should be forgotten, thereby extracting key contextual information. The GRU unit encodes information in a sequence through a series of mathematical operations, including reset gates, update gates, and calculation of candidate hidden states. These operations are implemented through a series of weight matrices and activation functions, enabling the GRU unit to adaptively learn and capture key features in the input sequence. In the encoder, the output hidden state values of the GRU unit are provided to the attention layer. The AM can dynamically adjust the corresponding weights based on the importance of different parts of the input sequence, thereby achieving further processing of the encoded data. In this way, the attention layer can weight the encoded data based on the attention level of contextual information, in order to obtain a more expressive and semantic representation. In summary, as the NN processing unit of the encoder, GRU obtains data from the input layer and provides the output hidden state values to the attention layer, thereby supporting further processing of the encoded data. It utilizes gating and AM to effectively extract and encode key information from input sequences, providing richer and more accurate inputs for subsequent task processing.

(3) Attention layer

The use of multi-attribute historical information as a training dataset is to fully utilize data with different attributes, as there are differences in importance between these attributes. This training dataset can contain multiple attributes such as network traffic, log information, system configuration, user behavior, etc., each of which provides different aspects of information. However, due to the fact that NN may treat these attributes equally when processing them, it is easy to overlook the importance of certain attributes. To address this issue, this article introduces a NN with AM. The NN with this AM helps to learn the interrelationships between various attributes, in order to gain a more comprehensive understanding of their impact on the current NS state. By placing the attention layer before the prediction network, it is possible to weight the security data of different attributes, thereby focusing more on the attributes that are more critical to the current prediction output. Through this approach, NN can be more focused and efficient in finding information that is more useful for predicting values, improving the accuracy of predictions. By introducing attention layers, NN can better process historical information of multiple attributes, avoiding treating different predictive factors equally and better understanding the importance of different attributes. This method can enable NN to better adapt to the correlation between different attributes and improve the quality of prediction results.

(4) Predictive network

This article can deliver the encoding vector obtained from the above steps to the prediction network to obtain the security situation value of the network at that time.

4.2. POS Algorithm Optimizes Model Optimization Parameters

GRU can greatly accelerate the training speed of NN and improve the efficiency of the algorithm by setting several appropriate hyperparameters. In practical applications, finding the optimal hyperparameters can not only improve the efficiency of algorithm operation but also enhance the accuracy of algorithm prediction. Therefore, finding the optimal hyperparameters is crucial for the entire model. Therefore, this article adopts the PSO algorithm [35] to optimize the model, and uses the algorithm's excellent global search ability to solve the hyperparameters required for the above model. The key points involved in the process of hyperparameter optimization include the number of particles, inertia weights, learning factors, and iteration times. A moderate number of particles can effectively improve search efficiency while avoiding excessive computational overhead. A

higher inertia weight helps to enhance the exploration ability of particles, thereby searching the solution space more widely, while a lower inertia weight helps particles converge to the local optimal solution faster, demonstrating the importance of inertia weight in search balance. The mechanism of dynamically adjusting learning factors can enable particles to adaptively change their search strategies based on the current local search situation, thereby improving overall optimization performance. By finding different optimal hyperparameter combinations based on different input data, the learning ability and prediction accuracy of NN can be improved [36].

In the minimization problem, the population consists of particles represented by vectors for velocity and direction:

$$V_{i,J}^{l} = \left[V_{i,1}^{l}, V_{i,2}^{l}, \ldots, V_{i,j}^{l}\right] \tag{5}$$

$$a_{i,J}^{l} = \left[a_{i,1}^{l}, a_{i,2}^{l}, \ldots, a_{i,j}^{l}\right] \tag{6}$$

In the formulas, j = 1, 2, 3, …, J, where J represents the spatial dimension of velocity and position vectors, i = 1, 2, 3, …, I, among them, I is the number of particles in the population, and l is the number of iterations.

Throughout the entire flight, the i-th particle found the optimal position $L_{i,J}^{l}$, and the optimal position found so far in the entire particle swarm is the best position g_{J}^{l} of the population, which can be expressed as Equations (7) and (8), respectively:

$$L_{i,J}^{l} = \left[L_{i,1}^{l}, L_{i,2}^{l}, \ldots, L_{i,j}^{l}\right] \tag{7}$$

$$g_{J}^{l} = \left[g_{1}^{l}, g_{2}^{l}, \ldots, g_{j}^{l}\right] \tag{8}$$

Update the optimal position solution for each particle based on the $L_{i,j}^{l}$ and g_{J}^{l} obtained from each iteration:

$$V_{i,J}^{p+1} = WV_{i,J}^{p} + c_1 * \text{rand} * \left(L_{i,j}^{l} - a_{i,j}^{pl}\right) + c_2 * \text{rand} - \left(g_{j}^{l} - a_{i,j}^{l}\right) \tag{9}$$

$$a_{i,J}^{l+1} = a_{i,J}^{l} + V_{i,J}^{l+1} \tag{10}$$

In the formula, the range of i and j is the same as Equation (5), rand is a random number between (0, 1), and W is the inertia weight; c_1 and c_2, respectively, represent the acceleration coefficients (learning factors) of the particle itself and social cognition, reflecting the degree to which the particle learns towards the optimal solution.

In standard PSO algorithms, the learning factors c_1 and c_2 are constant values. Due to the fact that the optimal solution in the population may change with each iteration, a constant value prevents particles from adaptively adjusting their position based on the optimal solution. Therefore, this chapter has made corresponding improvements to the PSO algorithm by setting c_1 and c_2 as variables, which can adaptively adjust the size based on the distance between particles and $L_{i,j}^{l}$ and g_{J}^{l}. The optimized learning factor adaptive adjustment formula is as follows:

Based on the global particle optimal solution and the optimal distance for adjusting the group position, this paper adopts c_1 and c_2 adaptive adjustment strategies:

$$c_1 = \frac{1}{1 + exp\left(-\left(L_{i,j}^{l} - a_{i,j}^{l}\right)\right)} \tag{11}$$

$$c_2 = \frac{1}{1 + exp\left(-\left(g_{i,j}^{l} - a_{i,j}^{l}\right)\right)} \tag{12}$$

In the formulas, $L_{i,j}^{l} - a_{i,j}^{l}$ represents the distance between the particle and the individual's optimal position at time l; $g_{i,j}^{l} - a_{i,j}^{l}$ represents the distance between the particle and the global optimal position at the l-th iteration.

The specific steps for obtaining the optimal parameters of the model through PSO algorithm after optimization are as follows:

(1) Initialize parameters.
(2) Randomly generate a group of three-dimensional particles representing the model's parameter combinations and initialize them.
(3) Use the performance indicators of the model as a fitness function: this function takes a parameter vector as input and returns the model performance corresponding to the combination of parameters. The model performance is negatively correlated with the performance fitness function.
(4) In step 3, the fitness of each individual is calculated using the fitness function in step 3. The fitness value of the current particle can be compared with the current optimal value. If the fitness value of the current particle is small, it can be updated. On this basis, this article searches for the optimal fitness point to become the global optimal solution.
(5) The velocity update formula of PSO can be used to update the position and velocity of each particle based on the individual optimal solution and global optimal solution in step 4, as well as some random factors.
(6) Determine whether the maximum number of iterations and fitness are close enough to the optimal solution to be met. If the conditions are met, proceed to step 7; otherwise, repeat steps 3–5 until the stopping conditions are met.
(7) Parameter vectors can be extracted as the optimal parameters for the model based on the parameters in step 6.
(8) The optimal parameters obtained can be applied to the model, retrained, and finally applied to practical problems. The specific process is depicted in Figure 3.

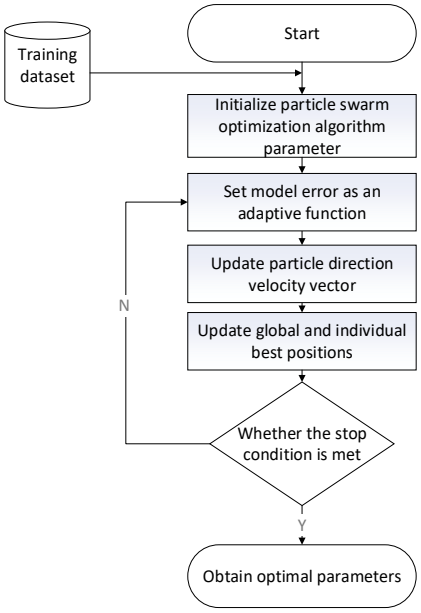

Figure 3. Specific flowchart of PSO algorithm.

5. Network Situation Experiment

5.1. Experimental Dataset

The UNSW-NB15 dataset can be used as experimental dataset 1. UNSW-NB15 includes various types of network attacks, covering various types of attacks from common to advanced, for the research and development of network intrusion detection systems. It

includes a mixture of normal and abnormal traffic, making it more realistic in simulating the actual network environment. The characteristics of diversity and authenticity help evaluate the adaptability of intrusion detection systems to various threats.

In this section of the experiment, the multi model method was validated using situational time series data from nearly 116 weeks, and the NS situation values were obtained by weighted average operation on the acquisition of NS situation elements and the frequency statistics of security events. The specific method is shown in Figure 4.

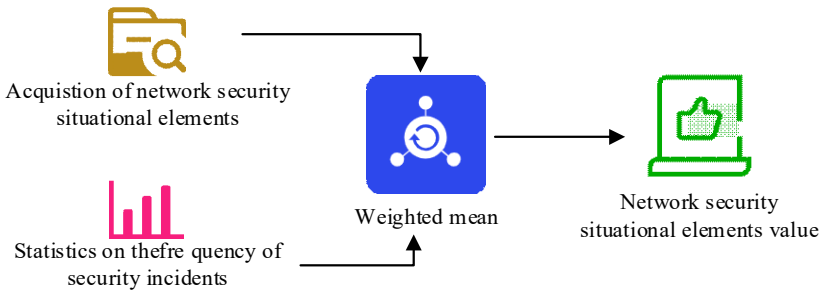

Figure 4. Specific method for evaluating NS situation values.

The normalized NS situation values are shown in Figure 5.

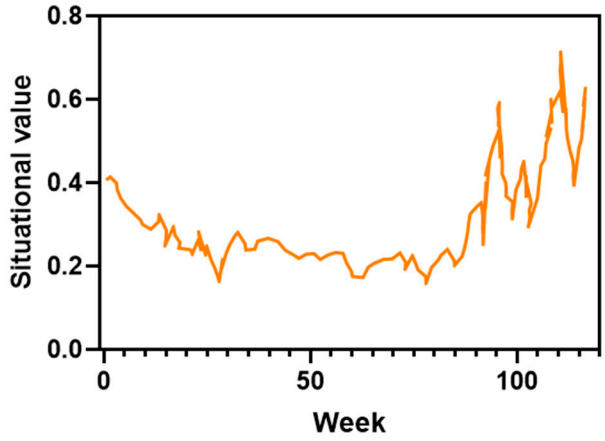

Figure 5. Normalized NS situation time series of UNSW-NB15 dataset.

Dataset 2 used the ADFA-IDS dataset, and during training the ADFA-IDS dataset was partitioned in the same way as the UNSW-NB15 dataset. Before preparing for the experiment, it is necessary to preprocess the security situation values of dataset 2 like dataset 1. The preprocessed ADFA-IDS NS situation values are shown in Figure 6.

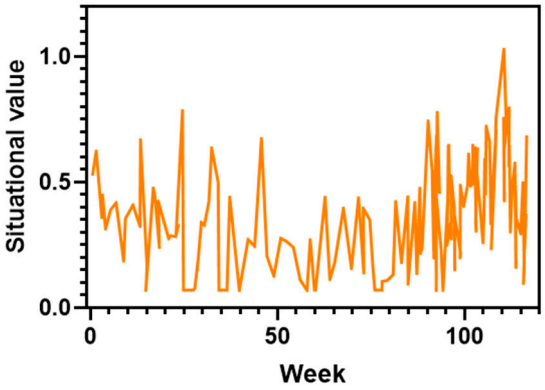

Figure 6. Normalized NS situation time series of ADFA-IDS dataset.

At the same time, in order to further ensure that the research method has sufficient generalization performance and to analyze the performance of the research method more deeply, experiments were conducted by adding the Information Security Operations Technology (ISOT) dataset and the Australian Defence Force Academy (ADFA) intrusion detection dataset. The ISOT dataset contains normal network traffic data and various types of attack data. The dataset provides multiple features, including timestamps, source IP addresses, target IP addresses, protocols used, packet sizes, workloads, and more. The ADFA intrusion detection dataset contains multiple real and synthetic types of network attacks, such as buffer overflow, DoS attacks, privilege escalation, etc. The data samples are labeled as normal or different types of attacks for supervised learning. The information and content examples of the dataset used in the study are shown in Table 2.

Table 2. Example of Dataset Information and Content.

Dataset Name	Description	Sample Size	Type	Key Features	Attribute Example
UNSW-NB15	Network traffic data with normal flow and various attacks.	2,540,044	Network Traffic	Timestamp, Source IP, Destination IP, Protocol, Packet Size, etc.	Timestamp: "1 October 2023 12:00:00"; Source IP: "192.168.1.1"; Attack Type: "DoS"
ADFA-IDS	Real and simulated network traffic data for intrusion detection.	100,000+	Network Traffic + Logs	Timestamp, Protocol Type, Source and Destination Addresses, etc.	Timestamp: "2 October 2023 16:00:00"; Source IP: "10.0.0.1"; Attack Type: "Privilege Escalation"
ISOT	Samples of traffic focused on network attack detection.	500,000+	Network Traffic	Timestamp, Source IP, Destination IP, Protocol Type, etc.	Timestamp: "3 October 2023 09:30:25"; Source IP: "172.16.0.1"; Attack Type: "Port Scan"
ADFA	Contains real and simulated network attack data.	Multiple Scenarios	Network Traffic + Logs	Timestamp, Attack Type, User Behavior, etc.	Timestamp: "4 October 2023 08:45:10"; Attack Type: "Buffer Overflow"; User Behavior: "Executing Shell"

5.2. Experimental Data Preprocessing

The selection of data from the first 90 weeks as the training set in this article means that the model can be trained using data from the past 90 weeks. This includes all observations and their related features from the past 90 weeks. This training set can be used to establish the model in this article and learn patterns and trends in the data. The data from the last 26 weeks can be used as a test set to evaluate the performance and accuracy of the trained model. This test set is data that the model has never been exposed to, so it can be used to simulate the performance of the model in real environments. A time window refers to dividing data into a series of continuous time periods, each of which is called a time window. The time step of GRU refers to the time interval between processing data during training and prediction. In this case, the time window can be set to the time step size of GRU, which means adjacent time steps can be used as input to train and predict the model. The network data with a prediction duration of one week each time means that the model can predict the trend and pattern for the next week based on previous observations and patterns. The duration of this prediction can be adjusted according to demand, but in this case, the data for one week of each prediction can be used as the output of the model.

5.3. Setting Evaluation Criteria and Setting Experimental Parameters

This chapter uses two error calculation methods, Mean Absolute Error (*MAE*) and Root Mean Square Error (*RMSE*), to evaluate the predictive performance of the model. The specific formulas are as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \widetilde{y_i}\right| \quad (13)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widetilde{y_i}\right)^2} \quad (14)$$

Firstly, this article preprocesses the experimental data, and then optimizes the prediction model using the hyperparameters of the PSO algorithm. It can first initialize the hyperparameters of the PSO algorithm, where the population particle count of the PSO algorithm is set to 5 and the number of iterations is 20. Meanwhile, to ensure a reasonable range of parameters, the value of the learning factor is set within the range of 0.5 to 2.2, with a weight factor of 0.8.

5.4. Multi Model Evaluation and Comparison

Three prediction models were trained on different datasets for MAE and RMSE. Due to the randomness of the weights in the NN model, the formula of calculating the average value through multiple experiments is adopted to increase accuracy. In model training, the hyperparameters of the network are optimized, with a learning rate set to 0.001, batch size set to 64, and training iterations set to 50 rounds. The model is trained using a cross entropy loss function and an Adam optimizer. During the training process, an early stopping mechanism is used to prevent overfitting, and the loss value of the validation set is monitored. The training is stopped when the validation set loss no longer decreases. The experiment was conducted in a GPU accelerated computing environment, using the TensorFlow/Kris deep learning framework to ensure efficient model training and testing. According to Tables 3 and 4, the Attention-CNN-BiGRU situational prediction method has certain advantages in performance. Compared with other predictions, the MAE and RMSE of this model are higher and the error is greater.

Table 3. Training losses of three models on dataset 1.

Network Model	MAE	RMSE
LSTM	0.1451	0.1698
GRU	0.1125	0.1286
Attention-CNN-BiGRU	0.0843	0.0932

Table 4. Training loss of the model on dataset 2.

Network model	MAE	RMSE
LSTM	0.0354	0.0605
GRU	0.0378	0.0512
Attention-CNN-BiGRU	0.0208	0.0296

This article compares the predictive accuracy of LSTM, GRU, and the Attention-CNN-BiGRU models in this article. Real value represents the actual situation value. The results are shown in Figures 7 and 8. All models have a certain level of predictive ability, most of which can predict approximate trend values, but the accuracy of each prediction model varies. According to Table 4, and Figures 7 and 8, it can be found that Attention-CNN-BiGRU has the smallest MAE and RMSE with Real value, and the line is more closely fitted. This indicates that the overall prediction accuracy of Attention-CNN-BiGRU is higher than that of LSTM and GRU. In order to further analyze the superiority of research methods, a comprehensive performance comparison was conducted by introducing Event trigging designed by Ye Z et al. [37], Representation learning approach designed by Wang Y et al. [38], and Federated Learning and Improved Transformer designed by Zhou Q et al. [39], as shown in Table 5.

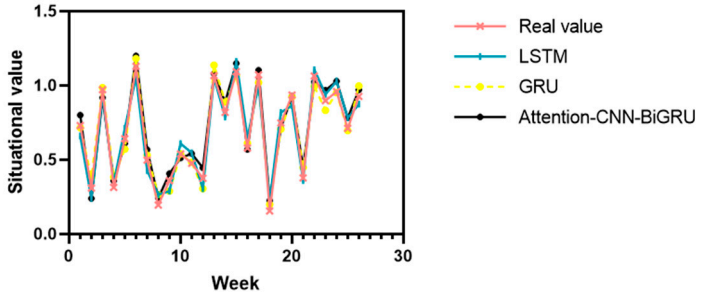

Figure 7. Comparison of situational values of different prediction models on the UNSW-NB15 dataset.

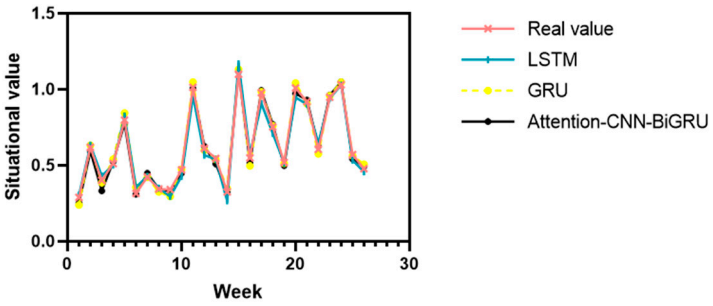

Figure 8. Comparison of situational values of different prediction models on the ADFA-IDS dataset.

Table 5. Comprehensive performance comparison.

Method	Data Set	Accuracy (%)	Recall Rate (%)	F1 Score (%)
Attention-CNN-BiGRU	ISOT	92.5	90	91.2
	ADFA	93.8	91.5	92.6
Event triggering	ISOT	88	85	86.4
	ADFA	87.5	83	85.2
Representation learning approach	ISOT	85.5	80	82.6
	ADFA	86	78.5	82.1
Federated Learning and Improved Transformer	ISOT	84	82	83
	ADFA	85.1	79	82

As shown in Table 5, on the ISOT dataset, the accuracy of Attention CNN BiGRU is 92.5%, the recall rate is 90.0%, and the F1 score is 91.2%. On the ADFA dataset, Attention CNN BiGRU has an accuracy of 93.8%, a recall rate of 91.5%, and an F1 score of 92.6%. On the ISOT dataset, the accuracy of Event triggering is 88.0%, the recall rate is 85.0%, and the F1 score is 86.4%. On the ADFA dataset, the accuracy slightly decreased, reaching 87.5%, the recall rate was 83.0%, and the F1 score was 85.2%. The performance of the representation learning approach is significantly lower than that of the research method, indicating its shortcomings in detection ability and reproducibility. The research method outperforms the other three methods in all indicators, indicating its outstanding performance in network security event detection. The high accuracy indicates that this method can accurately identify attack events, and the low false alarm rate improves the reliability of the security system. A high recall rate means that this method can capture a larger proportion of actual attacks and reduce the risk of false negatives. After introducing attention mechanism, the model can dynamically adjust the degree of attention to different features when processing input data. This means that the model can actively focus on features that are more important to the current situation, such as specific traffic patterns or abnormal use of protocols. This dynamic feature selection capability is crucial for improving model accuracy, as it allows the model to ignore a large amount of irrelevant information when determining the presence of attacks. The superior performance on multiple datasets also demonstrates the good generalization and adaptability of the research method.

5.5. Comparison of Time Complexity of Multiple Models

The time complexity of a model can represent its predictive efficiency, with lower time complexity leading to higher efficiency. Therefore, this article compares the prediction efficiency of different models by analyzing their training time complexity. The model parameters can be set: X samples and R neurons. The number of internal iterations during model training is represented by T = R = batchsize, and the model parameters are updated T times through iteration. There are two gate structures and one candidate state inside the GRU, and the attention layer is a feedforward NN. Therefore, the time complexity of this paper is $O(3T(2XR + R^2))$, and the complexity comparison with other prediction models is shown in Table 6.

Table 6. Model time complexity.

Network Model	Time Complexity
LSTM	$O(4T(2XR + R^2))$
GRU	$O(3T(2XR + R^2))$
Attention-CNN-BiGRU	$O(3T(2XR + R^2))$

According to Table 6, the time complexity of the Attention-CNN-BiGRU prediction model is the same as GRU and smaller than LSTM. Therefore, the efficiency of the Attention-CNN-BiGRU prediction model is the same as GRU and higher than LSTM.

6. Conclusions

The Attention-CNN-BiGRU NS situation prediction model adopts GRU to achieve NS situation prediction. While ensuring the efficiency of the prediction model is not affected, it improves the prediction accuracy by using multiple models for joint prediction, achieving a reduction in the loss of the prediction model during training iterations. Based on multi model performance comparison experiments, it is known that the Attention-CNN-BiGRU model has higher prediction accuracy for future NS than LSTM and GRU models. By comparing the time complexity of the three models, the Attention-CNN-BiGRU model has the same prediction efficiency as the GRU model and is higher than the LSTM algorithm. This fully demonstrates that the Attention-CNN-BiGRU model can improve the accuracy of model prediction while ensuring that the efficiency of the prediction model is not affected.

Author Contributions: Z.L.: methodology; X.L.: data curation; D.M.: investigation & funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China under grant 2021YFB3100901, NSF of China under Grant 62074131, 62272389, 62372069, and Shaanxi Provincial Key R&D Program 2023-ZDLGY-32.

Data Availability Statement: The data supporting the findings of this study are available within the article.

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

Glossary

NS	Network security
LSTM	Long Short-Term Memory
NN	Neural networks
BiGRU	Bidirectional Gated Recurrent Unit
GRU	Gated Recurrent Unit
PSO	Particle Swarm Optimization
DL	Deep learning
SW	Sliding window
HIDS	Hybrid Intrusion Detection System

IDS	Intrusion Detection System
RF	Random Forest
NIDS	Network Intrusion Detection System
Attention-CNN-GRU	Attention-Convolutional Neural Network Bidirectional Gated Recurrent Unit Neural Network
AM	Attention mechanism
RNN	Recurrent Neural Network
ISOT	Information Security Operations Technology
ADFA	Australian Defence Force Academy
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
z_t	the output of the update gate
W_z	the weights of the update gate
b_z	the biases of the update gate
$\overline{h}_t$	outputs between −1 and 1 by using the tanh function
J	the spatial dimension of velocity and position vectors
I	the number of particles in the population
L	the number of iterations
g_J^l	the best position
c_1	the acceleration coefficients (learning factors) of the particle itself
c_2	the acceleration coefficients (learning factors) of the particle social cognition
$L_{i,j}^l - a_{i,j}^l$	the distance between the particle and the individual's optimal position at time l
$g_{i,j}^l - a_{i,j}^l$	the distance between the particle and the global optimal position at the l-th iteration

References

1. Xue, R.; Tang, P.; Fang, S. Prediction of computer network security situation based on association rules mining. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 2794889. [CrossRef]
2. Zhang, R.; Liu, M.; Yin, Y. Prediction Algorithm for Network Security Situation based on BP Neural Network Optimized by SA-SOA. *Int. J. Perform. Eng.* **2020**, *16*, 1171.
3. Yang, H.; Zhang, L.; Zhang, X. An adaptive IoT network security situation prediction model. *Mob. Netw. Appl.* **2022**, *27*, 371–381. [CrossRef]
4. Tan, L.; Yu, K.; Ming, F.; Cheng, X.; Srivastava, G. Secure and resilient artificial intelligence of things: A HoneyNet approach for threat detection and situational awareness. *IEEE Consum. Electron. Mag.* **2021**, *11*, 69–78. [CrossRef]
5. Alavizadeh, H.; Jang-Jaccard, J.; Enoch, S.Y. A Survey on Cyber Situation-awareness Systems: Framework, Techniques, and Insights. *ACM Comput. Surv.* **2022**, *55*, 107. [CrossRef]
6. Zhu, Q. Research on road traffic situation awareness system based on image big data. *IEEE Intell. Syst.* **2019**, *35*, 18–26. [CrossRef]
7. Bi, B.; Huang, D.; Mi, B.; Deng, Z.; Pan, H. Efficient LBS security-preserving based on NTRU oblivious transfer. *Wirel. Pers. Commun.* **2019**, *108*, 2663–2674. [CrossRef]
8. Mahesh, B. Machine learning algorithms-a review. *Int. J. Sci. Res.* **2020**, *9*, 381–386. [CrossRef]
9. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
10. Kriegeskorte, N.; Golan, T. Neural network models and deep learning. *Curr. Biol.* **2019**, *29*, R231–R236. [CrossRef]
11. de Assis, M.V.O.; Carvalho, L.F.; Rodrigues, J.J.P.C. Near real-time security system applied to SDN environments in IoT networks using convolutional neural network. *Comput. Electr. Eng.* **2020**, *86*, 106738. [CrossRef]
12. Srinidhi, C.L.; Ciga, O.; Martel, A.L. Deep neural network models for computational histopathology: A survey. *Med. Image Anal.* **2021**, *67*, 101813. [CrossRef] [PubMed]
13. Hunter, J.; Porter, M.; Williams, B. Towards a theoretical framework for situational awareness in paramedicine. *Saf. Sci.* **2020**, *122*, 104528. [CrossRef]
14. 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]
15. Zhang, H.; Li, Y.; Lv, Z. A real-time and ubiquitous network attack detection based on deep belief network and support vector machine. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 790–799. [CrossRef]
16. Khraisat, A.; Gondal, I.; Vamplew, P. Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine. *Electronics* **2020**, *9*, 173. [CrossRef]
17. Gu, C. Research on prediction of investment fund's performance before and after investment based on improved neural network algorithm. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5519213. [CrossRef]
18. Han, X.; Liu, Y.; Zhang, Z.; Lu, X.; Li, Y. Network security posture prediction based on IFS-NARX model. *J. Jilin Univ. (Eng. Ed.)* **2019**, *49*, 273–279.

19. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
20. Van Houdt, G.; Mosquera, C.; Napoles, G. A review on the long short-term memory model. *Artif. Intell. Rev.* **2020**, *53*, 5929–5955. [CrossRef]
21. Boukhalfa, A.; Abdellaoui, A.; Hmina, N. LSTM deep learning method for network intrusion detection system. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 3315. [CrossRef]
22. Shewalkar, A.; Nyavanandi, D.; Ludwig, S.A. Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. *J. Artif. Intell. Soft Comput. Res.* **2019**, *9*, 235–245. [CrossRef]
23. Assis, M.V.O.; Carvalho, L.F.; Lloret, J. A GRU deep learning system against attacks in software defined networks. *J. Netw. Comput. Appl.* **2021**, *177*, 102942. [CrossRef]
24. Yang, J.; Shen, H.; Ge, P.; Dai, Y. Particle swarm optimization algorithm. *Software* **2020**, *3*.
25. He, C.; Zhu, J. A security posture prediction method of GRU neural network based on attention mechanism. *Syst. Eng. Electron. Technol.* **2020**, *43*, 258–266.
26. Xiong, W.; Legrand, E.; Åberg, O.; Lagerström, R. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Softw. Syst. Model.* **2022**, *21*, 157–177.
27. Georgiadou, A.; Mouzakitis, S.; Bounas, K.; Askounis, D. A cyber-security culture framework for assessing organization readiness. *J. Comput. Inf. Syst.* **2022**, *62*, 452–462. [CrossRef]
28. Rajadurai, H.; Gandhi, U.D. A stacked ensemble learning model for intrusion detection in wireless network. *Neural Comput. Appl.* **2022**, *34*, 15387–15395. [CrossRef]
29. Ferrari Dacrema, M.; Boglio, S.; Cremonesi, P.; Jannach, D. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Trans. Inf. Syst. (TOIS)* **2021**, *39*, 1–49. [CrossRef]
30. Cong, S.; Zhou, Y. A review of convolutional neural network architectures and their optimizations. *Artif. Intell. Rev.* **2023**, *56*, 1905–1969. [CrossRef]
31. Nguyen, M.T.; Huang, J.H. Fault detection in water pumps based on sound analysis using a deep learning technique. *Proc. Inst. Mech. Eng. Part E J. Process Mech. Eng.* **2022**, *236*, 298–307. [CrossRef]
32. Stupariu, M.S.; Cushman, S.A.; Pleşoianu, A.I.; Pătru-Stupariu, I.; Fuerst, C. Machine learning in landscape ecological analysis: A review of recent approaches. *Landsc. Ecol.* **2022**, *37*, 1227–1250. [CrossRef]
33. Kanagachidambaresan, G.R.; Ruwali, A.; Banerjee, D. Recurrent neural network. In *Programming with TensorFlow: Solution for Edge Computing Applications*; Springer: Cham, Switzerland, 2021; pp. 53–61.
34. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427. [CrossRef]
35. Roodschild, M.; Gotay Sardinas, J.; Will, A. A new approach for the vanishing gradient problem on sigmoid activation. *Prog. Artif. Intell.* **2020**, *9*, 351–360. [CrossRef]
36. Zhang, M.; Zhang, Y.; Cen, Q.; Wu, S. Deep learning–based resource allocation for secure transmission in a non-orthogonal multiple access network. *Int. J. Distrib. Sens. Netw.* **2022**, *18*, 15501329221104330. [CrossRef]
37. Ye, Z.; Zhang, D.; Cheng, J.; Wu, Z.G. Event-triggering and quantized sliding mode control of UMV systems under DoS attack. *IEEE Trans. Veh. Technol.* **2022**, *71*, 8199–8211. [CrossRef]
38. Wang, Y.; Liu, Z.; Xu, J.; Yan, W. Heterogeneous network representation learning approach for ethereum identity identification. *IEEE Trans. Comput. Soc. Syst.* **2022**, *10*, 890–899. [CrossRef]
39. Zhou, Q.; Wang, Z. A Network Intrusion Detection Method for Information Systems Using Federated Learning and Improved Transformer. *Int. J. Semant. Web Inf. Syst.* **2024**, *20*, 1–20. [CrossRef]