Hybrid and Spatiotemporal Detection of Cyberattack Network Traffic in Cloud Data Centers

Haitao Yuan, *Senior Member, IEEE*, Shen Wang, *Graduate Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—The rapid expansion of Internet users results in an immense influx of network traffic within extensive cloud data centers. Accurate and instantaneous identification and forecasting of network traffic aid system managers in efficiently distributing resources, assessing network performance based on specific service demands and scrutinizing the health of network status. However, sources and distributions of traffic are different, which makes accurate warnings of cyberattack traffic difficult. Recently, emerging neural networks have demonstrated their efficacy in forecasting time series data of network cyberattacks. The time series has temporal and spatial features, which can be efficiently captured with Informer and convolutional neural networks (CNNs). To realize high-performance spatiotemporal detection of cyberattacks, this work for the first time designs a hybrid and spatiotemporal prediction framework, which integrates CNNs, Informer, and a Softmax classifier to realize high-classification accuracy of normal and abnormal cyberattacks. Real-life data are adopted to evaluate the proposed method, which yields significant improvement in classification accuracy over typical benchmark classification models.**

***Index Terms*—Anomaly time series detection, deep learning, network cyberattacks, neural networks, spatiotemporal features.**

I. Introduction

COMPUTER networks are widely used in human daily lives and business [1]. In the process of computer network interactions, cloud data centers share software/hardware resources and use servers to exchange the network data, provide resources and information needed by the target, and realize network service functions, yielding a new network computing paradigm [2]. However, the growth of the scale of computer networks derives a variety of malicious attacks based on the network traffic. Table I shows some types of typical network attacks. Fig. 1 shows the relations between anomaly and attack types. For example, a Denial of Service (DoS) attack takes up too many resources of servers and hinders services for other legal users by sending requests beyond the normal range of demand [3]. Anomalies in the network traffic manifest as sudden surges within a short timeframe, resulting in significant disruptions to network service operations. Hence, it is necessary to proactively detect and prevent the attacks that threaten the network traffic services [4].

TABLE I
Network Attack Types

Methods of attacks	Description
DoS	An attack leading to extra resource usage, which negatively affects its original use.
Probe	A network attack collecting information about a network and checking its connected devices.
User to Root (U2R)	An attack that illegally accesses an account to manipulate critical resources of clients.
Remote to User (R2U)	An intrusion that gains user access on servers and attempts to request authorization for packet transmission.

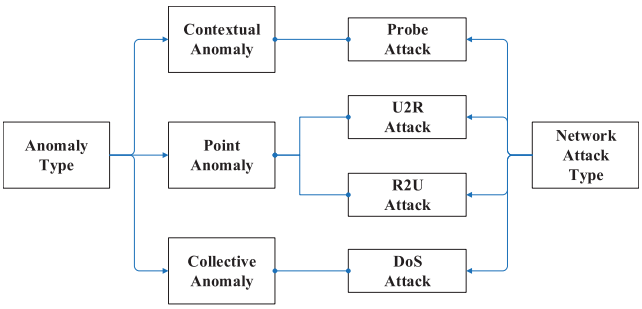


Fig. 1. Relations between anomaly and attack types.

Manuscript received 27 November 2023; revised 9 January 2024; accepted 26 January 2024. Date of publication 13 February 2024; date of current version 9 May 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62073005; in part by the Beijing Natural Science Foundation under Grant 4232049; and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015. This article was presented in part at the IEEE SMC 2023, Maui, HI, USA. *(Corresponding author: Haitao Yuan.)*

Haitao Yuan and Shen Wang are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn; 18376420@buaa.edu.cn).

Jing Bi is with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Jia Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75206 USA (e-mail: jiazhang@smu.edu).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/JIOT.2024.3360294

This work endeavors to preempt network attacks by promptly detecting incoming network traffic. It achieves this by discerning and categorizing anomalies rooted in traffic characteristics and anomaly types [7], [8]. In addition, the future attack traffic can be predicted based on the trend of the input historical traffic, and therefore, preventive measures can be taken in advance. Identifying attack traffic patterns within network servers' time series is a classification problem of univariate time series [9]. Traditionally, network traffic detection is mainly based on statistical methods. Discriminative features are extracted by analyzing the original data, and they are used by a classifier to distinguish normal and abnormal traffic [10]. However, due to the dramatic increase in the network scale, the network traffic often comes from multiple different service points, each of which usually does not have uniform characteristics and does not follow the same mathematical distributions [11]. Therefore, anomaly detection models based on distributed statistical learning often achieve relatively poor performance. They fail to yield the optimal learning models and capture hidden characteristics by manually designed models [12].

Deep learning provides automatic feature learning abilities and achieves better results than traditional methods in many fields [13]. Therefore, in recent years, several existing studies have applied deep learning methods to realize the traffic prediction, which captures hidden complex features from a large amount of the original network traffic through multiple iterations and accurately identifies the abnormal data [14], [15]. Compared with traditional statistical methods and machine learning algorithms, deep learning eliminates the process of manually designing features of the data set and does not require much expert knowledge. Thus, deep learning is suitable for network traffic detection in cloud data centers because it can well capture spatial and temporal changes over time [16]. Then, it can identify the normal and abnormal patterns in the network traffic data by utilizing the spatiotemporal patterns, thereby capturing network attack traffic changes. For example, deep learning techniques are adopted to classify and predict sensor data collected in industrial fields [17]. Different from existing studies, this work crafts a hybrid classifier leveraging spatiotemporal features inherent in network traffic data. It establishes deep learning models by amalgamating convolutional neural networks (CNNs) with Informer, achieving real-time detection and classification of network traffic anomalies. It focuses on the network traffic data with heterogeneous characteristics, mathematical distributions, and atypical and aperiodic changes [18].

The main contributions of this work are summarized as follows. First, this work preprocesses the network traffic data with a sliding window technique. Second, it extracts spatial features in each traffic window through a convolution layer and a pooling one in a CNN [19]. Third, temporal characteristics are captured in the output of the CNN layer by introducing Informer. Fourth, a Softmax classifier is adopted to classify the network traffic, thereby improving the detection ability to identify normal and abnormal traffic [20]. By integrating the above modules, this work proposes a hybrid and spatiotemporal detection framework named SCIS, which comprehensively combines a Sliding window mechanism, CNN, Informer, and a Softmax classifier.

The remaining of this work is given as follows. Section II gives the related studies. Section III illustrates the system architecture. Section IV shows details of the proposed method. Section V gives simulation experiments and results by using real-life data sets collected from real-life large-scale industrial clusters. Section VI concludes this work.

II. Related Work

This section shows the related work from two major perspectives, including cyberattack traffic detection models with temporal features and spatial ones, respectively.

A. Cyberattack Traffic Detection Models With Temporal Features

Traditional machine learning approaches depend on feature engineering to analyze cyberattack traffic detection. Abnormal traffic detection methods with deep learning can autonomously explore potential features in the data and find appropriate parameters through deep neural networks driven by big data [21]. Network traffic essentially comprises numerous substantial fluctuations over time. Certain researchers advocate for time series methodologies, such as long short-term memory (LSTM) [22] networks and recurrent neural networks (RNNs), which possess the capability to retain extensive the time series data and extract temporal features from the network traffic data [23]. Malhotra et al. [24] adopted sensor data containing normal signals to train LSTM models to predict future signals, and use real signals to calculate error distributions to achieve anomaly detection. This approach circumvents issues like gradient explosion and vanishing gradients encountered in RNNs, thereby expediting model convergence. Cheng et al. [25] adopted a sliding window algorithm to preprocess the traffic data for LSTM networks, balance the normal and abnormal data, and extract temporal features to yield satisfying results. Sainath et al. [26] proposed a two-stage LSTM model for structured information in network traffic, which takes advantage of the bidirectional learning characteristics of bidirectional LSTM. It obtains surface packet features and underly network flow ones, and integrates them into each unit for output, thereby obtaining more network traffic features and realizing accurate classification of traffic.

Deep learning models based on modeling of temporal features [27] mainly extract the temporal structure information of network traffic through the long-sequence dependence learning ability of the temporal network, which can learn periodic change characteristics of network traffic data and make periodic prediction [28], [29]. However, network attack traffic usually has atypical and aperiodic characteristics, reflected in values of single-point mutation. Time modeling models based on periodic changes for predicting abnormal data fail to handle them well. They cannot be used to analyze the actual mass data, including positive and negative samples [30].

B. Cyberattack Traffic Detection Models With Spatial Features

While RNNs enjoy widespread adoption, numerous studies have demonstrated that for certain sequence modeling tasks, the prediction accuracy of CNNs is comparable or, in some cases, even superior to that of RNNs [31]. Wang et al. [32] designed specific normal patterns of network traffic and achieve the identification of attack traffic by converting the time series into a 2-D image, and extracting its features through convolution operations. Ren and Wu [33] adopted a convolution algorithm to extract features from high-dimensional electroencephalogram signals and apply a deep concise network as a generative graphical model to classify ideal values. The methods extract spatial features by converting large-length temporal sequences and complex content distribution into images. Based on the 1-D sequence characteristics of network traffic, some researchers also adopt an 1-D convolutional network (Conv1D) [34]. Compared with 2-D graphics, the 1-D convolution can better learn and reflect the original characteristics of traffic data. Based on Conv1D, some researchers realize the traffic classification by integrating feature learning and selection learning into the network framework [35]. Specifically, after preprocessing the traffic, the model can automatically learn the nonlinear relationship, converge to the best point, and accurately and quickly realize the abnormal classification of traffic [36]. The method with spatial feature modeling extracts implicit spatial models of network traffic through CNNs, which can effectively capture spatial features of complex patterns in the sequence data [3]. However, traditional CNNs are unable to discover long-term dependencies. While extracting features from the sequence with CNNs, the loss of temporal information happens during operations of convolution and merging [37].

Due to the complexity of network traffic from different sources, models mentioned above cannot extract both temporal and spatial features of long-term network traffic. Therefore, this work proposes a prediction framework based on spatiotemporal features to improve the detection precision. Different from the study in [38], this work adopts a more advanced model of Informer instead of TCN to build the detection framework, and designs a hybrid optimization algorithm to optimize its key hyperparameters to improve the classification accuracy.

III. System Architecture

The abnormal traffic detection model based on spatiotemporal characteristics is derived from CNN and Informer [39]. The overall system process is given in Fig. 2. The specific data input and processing process can be summarized as follows.

1) To handle the problem of insufficient anomaly data proportion, SCIS preprocesses the input through a sliding window of fixed length to obtain the network traffic data samples.
2) SCIS extracts spatial features in the traffic window through a convolution layer and a pooling one in CNN and yields the intermediate output, which is shown in Fig. 3 for subsequent temporal feature extraction. Following the convolution and pooling layers, the intermediate output comprises a collection of data vectors.
3) SCIS extracts temporal features from the intermediate output through Informer. After that, the output of Informer is flattened into a feature vector used as the input to a Softmax classifier.
4) SCIS adopts a Softmax classifier to classify the data. The feature vector mentioned above is classified as either 0 or 1 by Softmax. Here, 0 indicates the label of normal samples, and 1 indicates that of abnormal samples.
5) The design of SCIS includes various model parameters, and its learning performance can be affected by these parameters. Therefore, SCIS adopts a hybrid optimization algorithm named genetic simulated annealing-based particle swarm optimization (GSPSO) to optimize its key hyperparameters to improve the classification accuracy. Here, this work takes an evaluation indicator of classification accuracy as the optimization target, and model hyperparameters to be optimized as decision variables of the optimization problem. Then, this work adopts GSPSO to solve it and the best found solution yields the best combination of model hyperparameters.

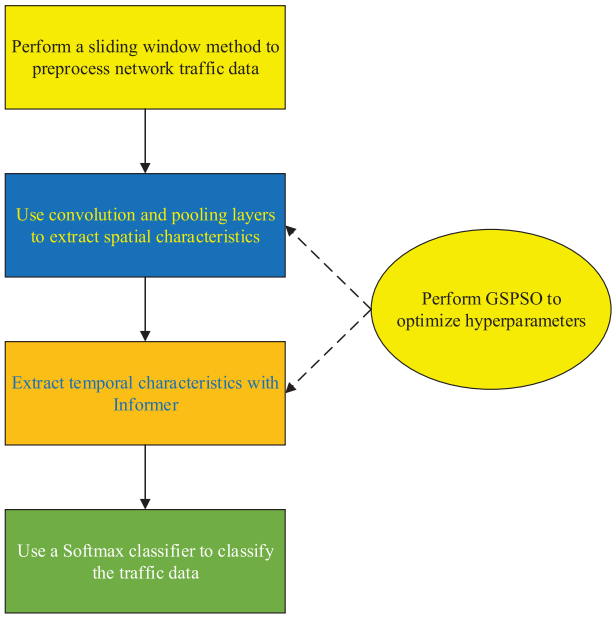


Fig. 2. System architecture. The model adopts Informer as a lower layer to learn temporal features. CNN and improved modules are used as an upper layer to learn spatial features, improve the detection ability for abnormal traffic, and enhance classification results of attack and normal traffic.

IV. Model Framework

This section describes the details of the abnormal traffic detection model. We first formulate our target problem and introduce a method of data processing. Then, we introduce the details of our model. Fig. 3 gives the framework for detecting anomalies in the network traffic and Table II lists main notations.

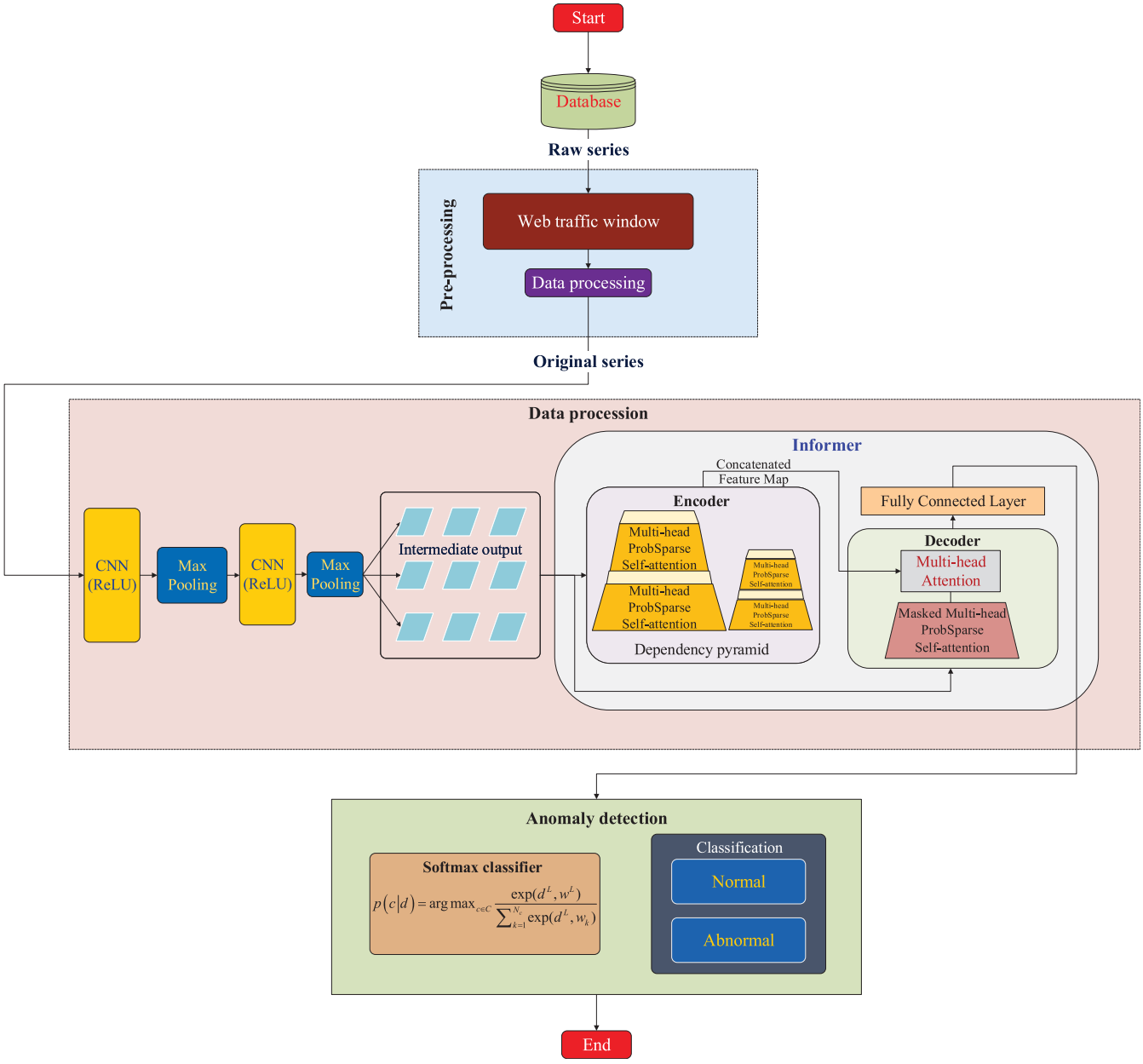


Fig. 3. Detection structure of the proposed SCIS.

A. Problem Definition

The abnormal traffic detection model is trained with network traffic data sets, including normal and abnormal labels [40]. The network traffic window sequence to be classified is the input. Through the neural network, the network weight values undergo updating, producing the corresponding label value for each segment as the output. The supervised learning process is repeated to obtain the best network parameters iteratively and achieve high-accuracy classification of network traffic sequences. The trained abnormal traffic detection model is capable of categorizing unclassified traffic data based on their sequential values. It can effectively assign labels to a substantial volume of original data, subsequently identifying and isolating abnormal traffic [41].

Let $X = \{x_1, \ldots, x_t, \ldots, x_T\}$ denote the time series data of network traffic spanning a duration of T, and $\tilde{X}$ ($\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_t, \ldots, \tilde{x}_T\}$) denotes the time series data of network traffic spanning a duration of T processed by the detection model. x_i and $\tilde{x}_i$ mean the sequence values of network traffic and the processed sequence traffic in time step i, respectively. l_T and $\tilde{l}_T$ denote the real label value and the classification label one corresponding to the window sequence with a time span of T, respectively [42]. This work classifies between normal and abnormal traffic series spanning a duration of T. The learning objective aims to establish the nonlinear mapping from the input sequence to its corresponding label value by minimizing the classification error function, $\text{loss}(\tilde{l}_T, l_T)$. The specific mathematical process is expressed as follows:

$$\tilde{l}_T = \Phi(\tilde{X}) \tag{1}$$

where Φ refers to a learned classifier function.

TABLE II
MAIN NOTATIONS

Notation	Definition
X	Sequence of network traffic
$\tilde{X}$	Sequence of network traffic after being processed
l_T	Real label value of a sequence
$\tilde{l}_T$	Classification label of a sequence
q	Input vector of traffic data
I	Dimension of the traffic data input vector
J	Convolution kernel number
L	Layer number of convolution
y_{ij}^{l}	Result of operating layer $l-1$ of dimension i in the feature value j
σ	Function of activation
b_j^{l-1}	Bias of feature map j in layer $l-1$
$W_{m,j}^{l-1}$	Weight coefficient of the j-th feature mapping kernel in dimension m of layer $l-1$
p^l	Maximum value of layer $l-1$
R	Pool size
$[.]_{\mathrm{AB}}$	Attention block
Conv1d(.)	1-D convolution layer
ReLU(.)	Activation function
MaxPool$(\cdot)$	Max pooling layer
$\boldsymbol{Q}_j^t$	Input of sequence t in layer j
$\boldsymbol{Q}_{de}^t$	Input of decoder of sequence t
$\boldsymbol{Q}_{\bigstar}^t$	Starting token
$\boldsymbol{Q}_0^t$	Placeholder for the target sequence
Concat(.)	Fully connected layer
$D_.$	Dimension of a sequence
d_*	Feature dimension after input representation
N	Particle number in each population
ζ	Mutation probability
c_1	Cognitive parameter
c_2	Social parameter
c	Acceleration parameter
T_0	Initial temperature
ϖ	Cooling coefficient
$\hat{\varsigma}$	Maximum value of inertia weight
$\check{\omega}$	Minimum value of inertia weight
f	Fitness value
$\check{\mathbf{x}}_i$	Locally best position of particle i
$\hat{\mathbf{x}}$	Globally best position of the population
$\hat{\mathbf{x}}_i$	Position of a superior particle for particle i
v	Velocity of particle

B. Data Processing

To address the data imbalance, the sliding window algorithm is employed [43]. It partitions the original data samples by utilizing a sliding window mechanism, i.e., in each step, a certain number of data points comprise a sample. If there is an outlier in the sample, the sample is labeled as abnormal; otherwise, it is normal.

C. CNN

In CNN, spatial features of the data are derived through a series of operations involving a convolution layer and a subsequent pooling one [44]. The convolution layer encompasses 1-D convolutions and pooling layers to autonomously extract spatial features from the network traffic sequence. Multiple filter vectors are applied in these convolution operations, sequentially traversing the sequence to unveil distinctive features. Following the convolution layer, an activation function is employed, enhancing its capacity to capture intricate features present in the input.

$q = (q_1, q_2, \ldots, q_p)$ denotes an input vector of traffic data. p denotes the size of each window. $q_z(1 \le z \le p)$ represents the normalized traffic data. I denotes the dimension of the traffic data input vector for this layer. i denotes the eigenvalue index $(1 \le i \le I)$. J denotes the convolution kernel number in this layer. j denotes the feature map index for each traffic window $(1 \le j \le J)$. The operation of the convolution layer is given as follows:

$$y_{ij}^{l} = \sigma\left(b_j^{l-1} + \sum_{m=1}^{M} W_{m,j}^{l-1} q_{i+m-1,j}^{l-1}\right) \tag{2}$$

where L denotes the layer number of convolution. y_{ij}^{l} denotes the output value derived from the convolution layer l $(1 \le l \le L)$. y_{ij}^{l} denotes the result of layer $l-1$ of dimension i in the feature value j from the map by using the value $q_{i+m-1,j}^{l-1}$ calculated from a layer of flow (the value of the input data q_{ij}^{0} is used if it is the first layer). $q_{i+m-1,j}^{l-1}$ denotes a traffic data vector of feature map j in dimension $i+m-1$ of layer $l-1$. b_j^{l-1} denotes the bias of feature map j in layer $l-1$. $W_{m,j}^{l-1}$ denotes the weight coefficient of the jth feature mapping kernel in dimension m of layer $l-1$. M denotes the filter size. σ denotes the function of activation (such as tanh or ReLU). The pooling layer is performed as follows:

$$p^l = \mathbf{max}_{r \in R} y_{i \times L + r, j}^{l-1} \tag{3}$$

where $y_{i \times L + r, j}^{l-1}$ means the value of feature map j in dimension $i \times L + r$ of layer $l-1$. R denotes the pool size and R is smaller than the input size y. L denotes the step length to move the pool area. p^l denotes the maximum value of layer $l-1$.

The maximum pool scans the feature map in steps, selects the maximum value, and outputs it to the next layer. Following maximum pooling, the dimension of a feature map is reduced by half in width and height, while retaining the same number of channels. This achieves dimensionality reduction and compresses the feature map, consequently decreasing the network's parameter number and the computational complexity. Additionally, this process helps mitigate overfitting issues [45].

D. Informer Model

Different from traditional RNNs, LSTM, gate recurrent unit (GRU), and other models, Informer adopts an improved attention mechanism, which deals with long-term dependencies and missing values in sequences as well. Key characteristics of Informer are given as follows. It employs a multiscale-time structure of encoder and decoder, enabling simultaneous consideration of information across various time scales. Besides, it uses an attention mechanism of adaptive length, which

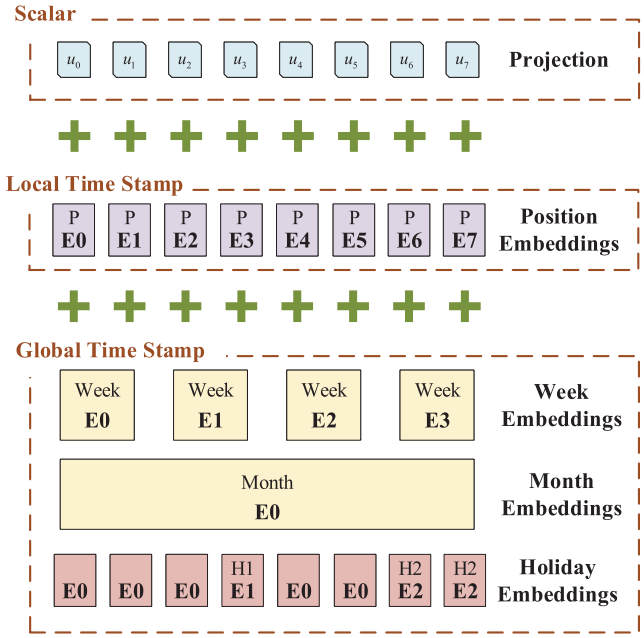


Fig. 4. Input representation of Informer. The input embeddings include three individual components, i.e., scalar projection, local time stamp embeddings, and global time stamp embeddings.

can automatically adjust the attention range according to the sequence length, thereby handling long sequences well. Then, a novel gated convolution unit is used to decrease the number of parameters and computations while improving the generalization ability. It also handles missing values in sequences as well by using a masking mechanism that automatically handles missing values during the training.

In the problem of long sequence modeling, local and hierarchical timing information, e.g., week, month, and year, and sudden timestamp information, e.g., events or holidays can be investigated. Traditional self-attention mechanisms may lead to the mismatch of keys and queries between the encoder and decoder, and ultimately influence the prediction performance. However, the Informer can avoid this problem, and its input representation is given in Fig. 4.

In the encoder module, the Informer extracts the most important attention information by the distilling operation. It first designs multiple stacks, each with input representations of the above-mentioned tokens and timestamps. Then, the input passes through a layer of Conv1d convolution to yield a representation of size $L \times d$ and adds them to the input of the attention block. Then, the representation passes through multiple attention blocks, each with multiple ProbSparse self-attention modules. After the output of each block, there is a sequence of operations, including a Conv1d convolution layer, a ReLU activation layer, and a maxpooling layer with a step size of 2. The specific formula is given as

$$\boldsymbol{Q}_{j+1}^{t} = \text{MaxPool}\Big(\text{ReLU}\Big(\text{Conv1d}\Big(\Big[\boldsymbol{Q}_{j}^{t}\Big]_{\text{AB}}\Big)\Big)\Big) \tag{4}$$

where $[.]_{\text{AB}}$ denotes an attention block, Conv1d$(\cdot)$ denotes a 1-D convolution layer on time dimension with the activation function of ReLU, MaxPool$(\cdot)$ denotes a max pooling layer, and $\boldsymbol{Q}_{j}^{t}$ denotes the input of sequence t in layer j. Consequently, it consolidates the outputs from multiple stacks and derives the final output of the encoder. In the decoder module, Informer adopts a decoder similar to the traditional one. To generate the output of a long sequence, the following input is given as:

$$\boldsymbol{Q}_{de}^{t} = \text{Concat}\Big(\boldsymbol{Q}_{\bigstar}^{t}, \boldsymbol{Q}_{0}^{t}\Big) \in \boldsymbol{R}^{(D_{\bigstar}+D_{y})\times d_{*}} \tag{5}$$

where $\boldsymbol{Q}_{de}^{t}$ denotes the input of decoder of sequence t, Concat$(\cdot)$ denotes a fully connected layer, $\boldsymbol{Q}_{\bigstar}^{t}$ denotes the starting token, $\boldsymbol{Q}_{0}^{t}$ denotes a placeholder for the target sequence, $D_{\bigstar}$ denotes the dimension of $\boldsymbol{Q}_{de}^{t}$, D_y denotes the dimension of $\boldsymbol{Q}_{0}^{t}$, and d_* denotes the feature dimension after input representation. The first component is the starting token sequence, and the second component is the part that needs to be predicted where the scalar is filled with 0. The sequence undergoes a masked ProbSparse self-attention layer and subsequently progresses through a fully connected layer to yield the ultimate output.

E. GSPSO Algorithm

This work adopts an improved optimization algorithm named GSPSO [46] to optimize the hyperparameters in SCIS. Currently, there are various typical optimization methods to solve the hyperparameter optimization problem, but those algorithms all have their shortcomings. For example, genetic algorithm (GA) has high diversity of individuals, and simulated annealing (SA) has high-global search ability, but they both have slow convergence. particle swarm optimization (PSO) is known for its rapid convergence, yet it tends to prematurely converge to local optima. Thus, we propose GSPSO that combines their strengths. GSPSO incorporates the rule of Metropolis acceptance in SA and genetic manipulations in GA to enhance the performance of PSO. Compared with traditional PSO, each particle in GSPSO possesses a distinguished superior particle that excels among the population. In addition, this superior particle plays a guiding role in shaping the collective search process of the entire population.

Let $\acute{\mathbf{x}}_i$ represent the position of a superior particle designated for each individual particle i, while $\hat{\mathbf{x}}$ signifies the globally best position for the entire population. $\acute{\mathbf{x}}_i$ is designed as a combination of $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ with

$$\acute{\mathbf{x}}_i = \frac{c_1\theta_1\check{\mathbf{x}}_i + c_2\theta_2\hat{\mathbf{x}}}{c_1\theta_1 + c_2\theta_2} \tag{6}$$

where c_1 and c_2 represent two parameters revealing influences of $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$, and θ_1 and θ_2 mean two random constants uniformly generated in $(0, 1)$. Velocities of particles v_i and positions of particles $\mathbf{x}_i^{\varsigma+1}$ are updated as

$$v_i = \omega \cdot v_i + c \cdot \mu \cdot \big(\acute{\mathbf{x}}_i - \mathbf{x}_i^{\varsigma}\big) \tag{7}$$

$$\mathbf{x}_i^{\varsigma+1} = \mathbf{x}_i^{\varsigma} + v_i \tag{8}$$

where μ denotes a vector comprising randomly generated numbers uniformly distributed in the range of (0, 1). If $f(\mathbf{x}_i^{\varsigma+1}) \leq f(\mathbf{x}_i^{\varsigma})$, $\mathbf{x}_i^{\varsigma+1}$ is selected; otherwise, it is conditionally selected if

$$e^{\left(\frac{f(\mathbf{x}_i^{\varsigma}) - f(\mathbf{x}_i^{\varsigma+1})}{T_{\varsigma}}\right)} > \xi \tag{9}$$

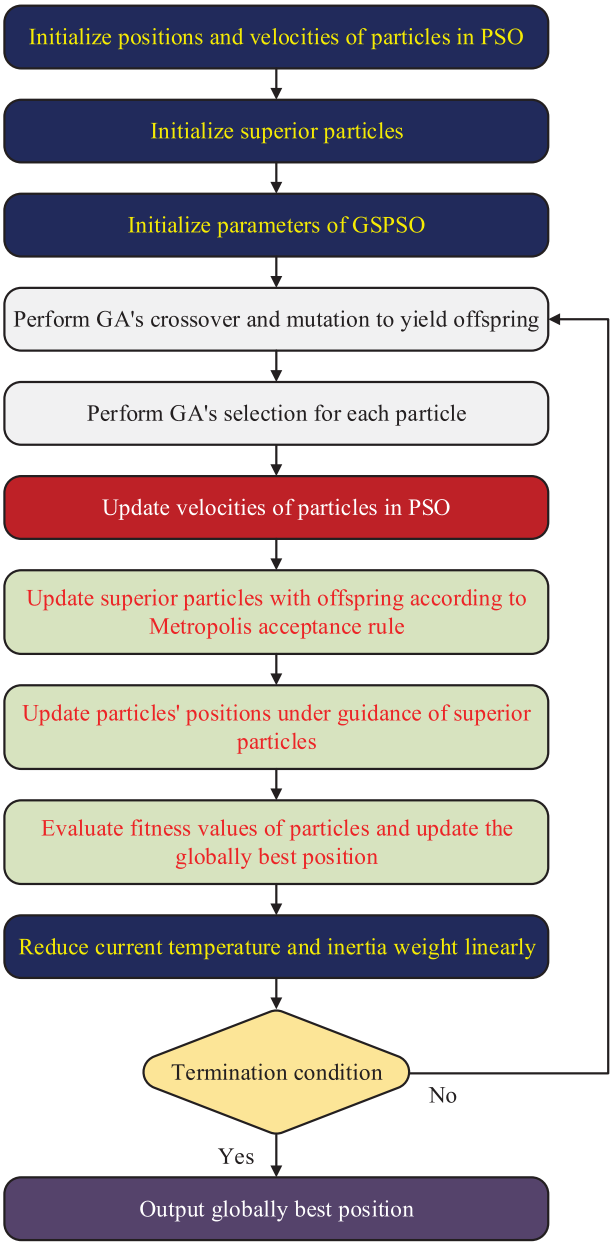


Fig. 5. Detailed process of GSPSO.

where T_ς signifies the temperature in iteration ς and ξ represents a randomly generated number uniformly distributed in the range of (0, 1).

The detailed process of GSPSO is given in Fig. 5 and the pseudo codes of GSPSO are shown in Algorithm 1. Line 1 shows that the random initialization of both particle positions and velocities occurs. Line 2 calculates the fitness (f) of each particle. Line 3 updates $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$. Line 4 sets parameters of SA, GA, and PSO. Here, N represents the number of particles in each population, ζ represents the mutation probability, c_1 and c_2 denote cognitive and social parameters, c denotes the acceleration parameter, T_0 denotes the initial temperature, ϖ denotes the cooling coefficient, $\hat{\varsigma}$ denotes the number of iterations, $\hat{\omega}$ denotes the maximum value of inertia weight and $\check{\omega}$ denotes the minimum one. Line 6 means that when

Algorithm 1 GSPSO

1: Initialize velocities and positions of particles randomly
2: Update fitness values (f) of particles
3: Update the locally best position ($\check{\mathbf{x}}_i$) of each particle i, and the globally best position ($\hat{\mathbf{x}}$) of the population
4: Initialize ζ of GA, T_0 and ϖ of SA, and PSO's parameters including c_1, c_2, c, $\hat{\omega}$, $\check{\omega}$, $\hat{\varsigma}$, and N
5: $\varsigma \leftarrow 1$
6: **while** $\varsigma \leq \hat{\varsigma}$ **do**
7: Apply GA's crossover operation on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ to produce an offspring $\check{\mathbf{x}}_i$
8: Apply GA's mutation operation on each bit of the offspring $\check{\mathbf{x}}_i$ with a certain probability ζ
9: Apply selection operation to pick out $\check{\mathbf{x}}_i$ or $\acute{\mathbf{x}}_i$
10: Update v_i^ς in PSO with (7)
11: Update $\mathbf{x}_i^{\varsigma+1}$ with (8)
12: Selectively update $\mathbf{x}_i^{\varsigma+1}$ with (9)
13: Calculate f of particles
14: Update $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$
15: $T_\varsigma \leftarrow T_{\varsigma-1}\varpi$
16: $\omega \leftarrow \hat{\omega} - \frac{\varsigma(\hat{\omega}-\check{\omega})}{\hat{\varsigma}}$
17: $\varsigma \leftarrow \varsigma + 1$
18: **end while**
19: **return** $\hat{\mathbf{x}}$

TABLE III
PARAMETER SETTING OF GSPSO

N	ζ	$c_1(c_2)$	c	T_0	ϖ	$\hat{\varsigma}$	$\hat{\omega}$	$\check{\omega}$
40	0.05	0.5(0.5)	1.5	10^8	0.95	2000	0.95	0.4

$\varsigma \leq \hat{\varsigma}$, the while loop terminates. Line 7 yields $\check{\mathbf{x}}_i$. Line 8 applies the mutation operation to $\check{\mathbf{x}}_i$ with ζ. Line 9 shows whether $\check{\mathbf{x}}_i$ is picked out with the selection operation. Lines 10 and 11 change velocities and positions with (7) and (8), respectively. Line 12 selectively updates positions with (9). Line 13 calculates fitness values of particles. Line 14 changes $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$. Line 15 decreases T_ς by ϖ. Line 16 decreases ω in each iteration ς sequentially from $\hat{\omega}$ to $\check{\omega}$. Line 19 outputs the best $\hat{\mathbf{x}}$ that have been found.

In this work, model hyperparameter values and evaluation indicators are positions and fitness values, respectively. For instance, we regard the mean absolute error (MAE) as a fitness value and the kernel size as a position, and each kernel size has its corresponding MAE. Thus, GSPSO can find the globally best position with the corresponding best fitness value, which is the best value of model hyperparameter. There are a few variables involved in this optimization problem, and the parameter setting of GSPSO is not particularly elaborate. We set our algorithm parameters according to [46]. The primary parameter settings are exhibited in Table III.

V. EXPERIMENTAL RESULTS

This work designs several simulations to illustrate the performance of SCIS and compares it with its state-of-the-art peers. This section describes our experiments and analyzes the results.

TABLE IV
EXAMPLE OF DATAFRAME AFTER DATA PREPROCESSING

0	1	2	3	...	57	58	59	60
0.0239	0.0449	0.0590	0.0459	...	0.0085	0.0143	0.0178	0
0.0449	0.0590	0.0459	0.0236	...	0.0143	0.0178	0.0137	0
0.0590	0.0459	0.0236	0.0220	...	0.0178	0.0137	0.0062	0
0.0459	0.0236	0.0220	0.0178	...	0.0137	0.0062	0.0193	0

TABLE V
MODEL PARAMETERS OF SCIS

Type	Filter	Kernel Size	Stride	Output Shape
Conv1D_1	64	5	1	(60, 64)
Activation (ReLU)	-	-	-	(60, 64)
MaxPooling1D_1	-	2	2	(30, 64)
Conv1D_2	64	5	1	(30, 64)
Activation (ReLU)	-	-	-	(30, 64)
MaxPooling1D_2	-	2	2	(15, 64)
Reshape	-	-	-	(960, 1)
Informer	-	-	-	(64, 1)
Flatten	-	-	-	(64)
Dense(32)	-	-	-	(32)
Activation (ReLU)	-	-	-	(32)
Dense(2)	-	-	-	(2)
Softmax	-	-	-	(1)

A. Yahoo S5 Webscope Data Set

This work adopts the data collected from a US portal site of Yahoo. We adopt the A1 class of anomaly benchmark data set of Yahoo Webscope S5 to train and test the proposed SCIS. The data is presented as a time series of traffic derived from real-life network services with a 1 hr sampling interval. Outliers are manually flagged and have relatively large changes in traffic compared with other available data sets. The input values for our SCIS model are in (0, 1), and we preprocess and normalize them for anomaly detection with

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{10}$$

where x denotes a value of the original traffic, $x_{\min}$ denotes the minimum value, $x_{\max}$ denotes the maximum value, and x' denotes its normalized one.

There are 94 866 traffic values in all 67 files for class A1, where only 1669 of these values are abnormal. The ratio of outliers in the data used is very small, i.e., only 1.7%. Thus, to address the data imbalance issue, we employ a sliding window mechanism with a step size of 60 to aggregate the original data samples. This signifies that each sample comprises 60 consecutive flow points. If an outlier exists within the sample, it is labeled as abnormal; otherwise, it is labeled as normal. Then, a window slides to the next flow point to form the next sample. After processing, 88 726 samples are obtained, of which 8473 are abnormal samples with an abnormal proportion of 9.5%.

Table IV shows the first four rows of the datafame after data preprocessing. The first 60 rows are listed as the traffic values in a single sample, and the 61st row is listed as the label value. Fig. 6 gives the normalized distribution of the network traffic. The model parameters of SCIS are exhibited in Table V.

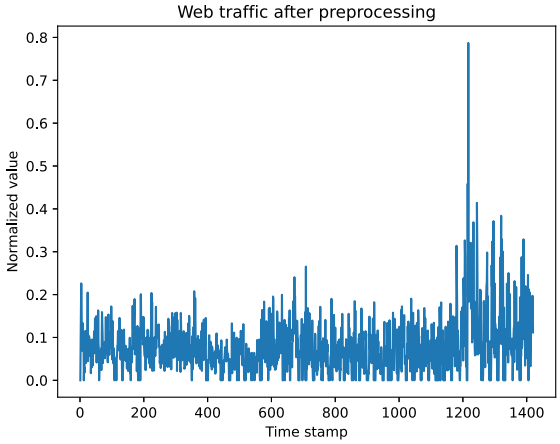


Fig. 6. Web traffic after preprocessing. The horizontal axis is the time stamp and the vertical one is the standardized value of traffic. The significant differences between the outliers and other normal traffic points are shown in this figure.

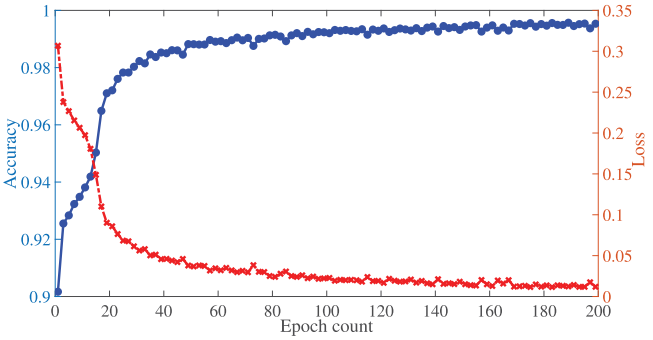


Fig. 7. Changes of accuracy and loss in each epoch.

TABLE VI
CONFUSION MATRIX

GT \ PR	Normal	Abnormal
Normal	23479	507
Abnormal	661(♠)	1971(♢)

B. Model Training Results

The traffic detection model is used for training, and the parameters of training are given as follows. The size of batch is 512 and the number of epochs is 200. The preprocessed A1 training set is utilized for training, maintaining a training-to-test set ratio of 7:3. Fig. 7 shows the change process of model accuracy and loss in the iterative training process of 200 epochs. It is observed that our SCIS model yields stable and high–accuracy results after 100 epochs.

The CPU and memory utilization are extracted and aggregated for different training intervals, e.g., 5 mins., 10 mins., ..., 60 mins. Figs. 8 and 9 show the comparison of the memory and CPU utilization of four selected methods, respectively.

C. Analysis of Misclassification Data

The substantial data imbalance in anomalous traffic detection underscores the importance of precisely identifying a limited number of abnormal instances. Table VI shows the

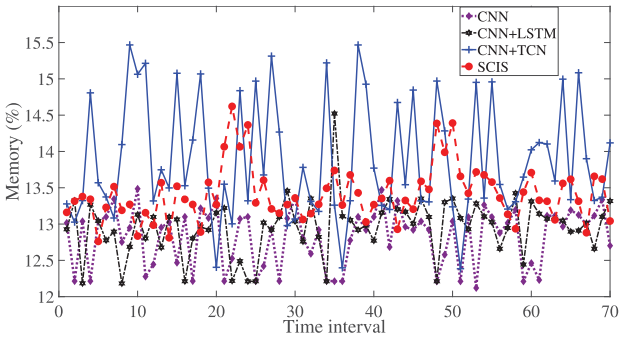


Fig. 8. Memory utilization for four selected methods.

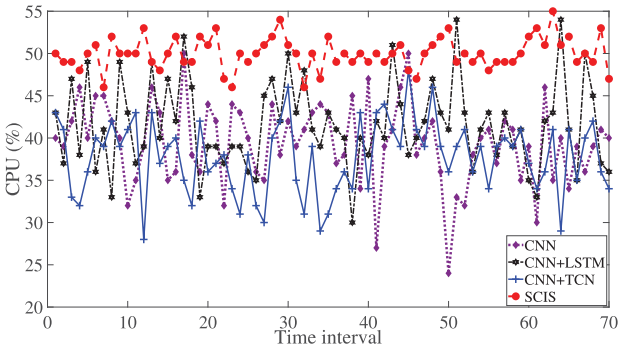


Fig. 9. CPU utilization for four selected methods.

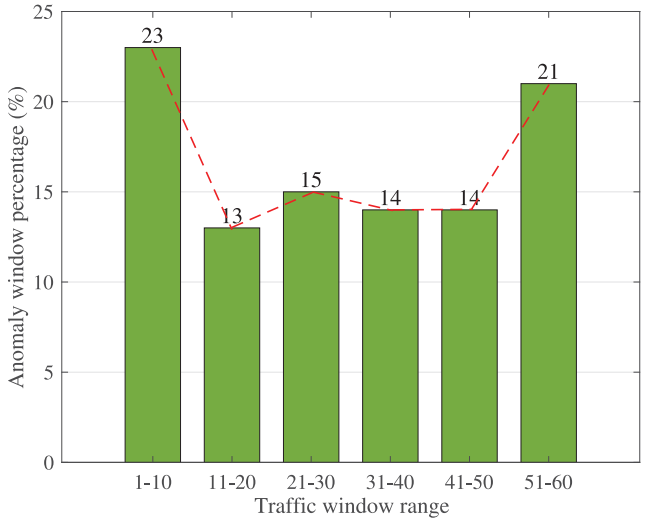


Fig. 10. Percentage distribution of misclassifications for different positions of anomaly values.

confusion matrix in one round of prediction and the classification results of the testing data with the trained model.

Here, the symbol ♠ shows the number of classification errors. The ground truth (GT) is abnormal but its prediction result (PR) is normal. The symbol ◇ denotes the number of sliding traffic windows correctly identifying anomalies, while ♠ is crucial for analyzing the characteristics of misclassifications. Precisely, traffic windows comprising 60 data points are segmented into six parts, categorized according to the locations of anomalies within each window. Fig. 10 illustrates the percentage of misclassification results corresponding to

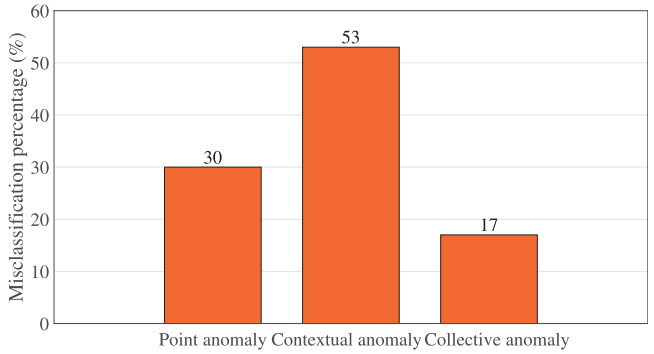


Fig. 11. Percentage of point anomalies, contextual ones, and collective ones not detected by SCIS.

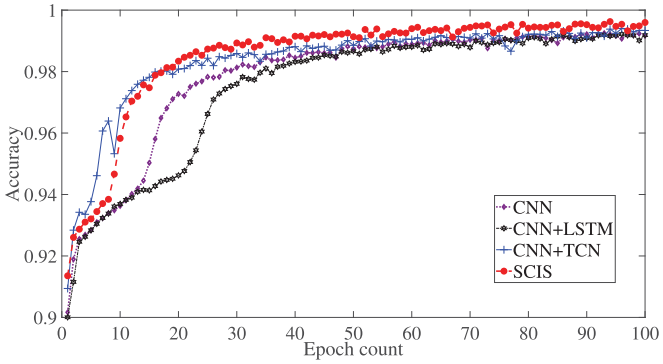


Fig. 12. Accuracy in each epoch.

each abnormal position concerning ♠. If an abnormal position resides at either end of a traffic window, it tends to induce misclassifications. Conversely, when the abnormal value is situated at the center, a reduction in the number of misclassifications is observed. We additionally depict the categories of anomalies present in ♠. Fig. 11 illustrates the distribution of percentages of point anomalies, contextual ones, and collective ones undetected by SCIS. It is evident that detecting contextual anomalies poses the greatest challenge, followed by point ones and collective ones, respectively.

D. Comparison of Algorithms

Figs. 12 and 13 present the accuracy and loss of four models in each epoch. SCIS yields the highest accuracy and the lowest loss, and it is observed that the abnormal detection indicators are improved. An error occurs when an abnormal instance is identified as normal (false negative, F_N), or when a normal instance is identified as abnormal (false positive, F_P). Likewise, a true positive (T_P) sample happens when a normal instance is correctly classified, while a true negative (T_N) sample happens when an abnormal instance is accurately identified. The performance of each algorithm is evaluated with T_P, T_N, F_P, and F_N, respectively

$$\check{P} = \frac{T_P}{T_P + F_P} \tag{11}$$

$$\check{R} = \frac{T_P}{T_P + F_N}. \tag{12}$$

TABLE VII
COMPARISON OF TRAINING RESULTS OF FOUR MODELS FOR THE YAHOO DATA SET

Steps	CNN			CNN+LSTM			CNN+TCN			**SCIS**		
	$\check{P}$	$\check{R}$	$\check{F}$	$\check{P}$	$\check{R}$	$\check{F}$	$\check{P}$	$\check{R}$	$\check{F}$	$\check{P}$	$\check{R}$	$\check{F}$
1	0.6356	0.6462	0.6409	0.6870	0.5930	0.6366	0.7654	0.6622	0.7101	**0.7753**	**0.6888**	**0.7294**
2	0.6154	0.6519	0.6331	0.7080	0.5759	0.6352	0.7634	0.7074	0.7343	**0.7953**	**0.7488**	**0.7714**
3	0.6770	0.6086	0.6410	0.7234	0.6003	0.6561	0.7132	0.6861	0.6994	**0.8093**	**0.6954**	**0.7480**
4	0.6257	0.6371	0.6314	0.6640	0.5873	0.6233	0.7245	0.6466	0.6833	**0.7842**	**0.6934**	**0.7360**

TABLE VIII
COMPARISON OF EVALUATION INDICATORS OF FOUR MODELS FOR THE YAHOO DATA SET

Steps	CNN			CNN+LSTM			CNN+TCN			**SCIS**		
	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	$\boldsymbol{R^2}$	**RMSE**	**MAE**
1	0.2613	0.2676	0.0716	0.2858	0.2587	0.0669	0.3009	0.2512	0.0534	**0.3798**	**0.2272**	**0.0516**
2	0.2844	0.2733	0.0747	0.3048	0.2557	0.0654	0.3161	0.2349	0.0506	**0.3906**	**0.2094**	**0.0438**
3	0.2570	0.2596	0.0673	0.2959	0.2494	0.0622	0.3130	0.2436	0.0583	**0.3870**	**0.2236**	**0.0499**
4	0.2765	0.2712	0.0735	0.2799	0.2649	0.0701	0.3309	0.2305	0.0567	**0.3859**	**0.2156**	**0.0487**

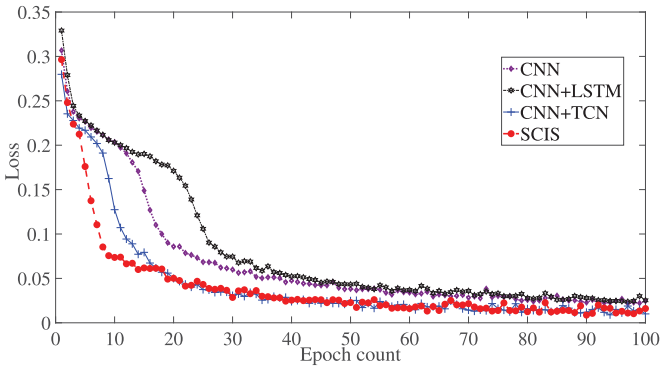


Fig. 13. Loss function value in each epoch.

In (11), $\check{P}$ representing precision signifies the proportion of relevant instances among all retrieved ones. In (12), $\check{R}$ representing recall denotes the percentage of relevant instances retrieved among all relevant ones. $\check{R}$ serves as a crucial metric for anomaly detection as it represents the ratio of the detected abnormalities to the total number of abnormal instances. Based on $\check{P}$ and $\check{R}$, $\check{F}$ denoting F1 score is derived with (13). This work quantitatively shows the results of evaluation metrics

$$\check{F} = 2\frac{\check{P} \times \check{R}}{\check{P} + \check{R}}. \tag{13}$$

Table VII shows the comparative results of the above metrics. CNN is used as a baseline and all experiments are performed in each combination of improved models. SCIS has a composite structure and optimized hyperparameters, and therefore, it achieves higher performance than other models with respect to $\check{P}$, $\check{R}$, and $\check{F}$. To evaluate the performance of SCIS, we select coefficient of determination (R^2), MAE, and root mean square error (RMSE) to compare it with three typical models. Table VIII shows the comparison of evaluation indicators of SCIS and the other three models for the Yahoo data set. The selected indicators show the deviations between predicted results and GT ones. It is observed that SCIS yields higher larger accuracy of prediction than others with respect to R^2, RMSE, and MAE.

TABLE IX
EVALUATION INDICATORS WITH DIFFERENT VALUES OF KERNEL SIZE

Kernel size	R^2	RMSE	MAE
5	**0.3858**	**0.2190**	**0.0485**
10	0.1727	0.2714	0.0737
15	0.1869	0.2808	0.0633
20	0.3455	0.2322	0.0594

TABLE X
EVALUATION INDICATORS WITH DIFFERENT VALUES OF D_FF

d_ff	R^2	RMSE	MAE
128	0.3043	0.2489	0.0619
256	**0.3858**	**0.2190**	**0.0485**
512	0.3514	0.2310	0.0588
1024	0.3124	0.2593	0.0579

SCIS optimizes its several key parameters with the proposed GSPSO. Table IX shows the results with MAE as the optimization objective function and the kernel size in CNN as an optimization variable. It is shown from the results that MAE is the smallest when the kernel size is five. Table X shows the results with MAE as the optimization objective function and d_ff in Informer as an optimization variable. It is shown from the results that MAE is the smallest when the d_ff is 256.

VI. CONCLUSION

The network attacks may cause serious damage to network service operations, and therefore, it is necessary to proactively detect and prevent them from threatening network traffic services. This work proposes a hybrid and spatiotemporal network traffic detection method by well combining a sliding window, CNN, informer, and a Softmax classifier (SCIS). It takes into account the impact of various sources and traffic distribution patterns and effectively attains the merits of traffic detection models based on temporal features and those based on spatial features. Specifically, SCIS adopts the sliding window to handle the issue of unbalanced data samples. CNN is adopted to enhance its capability in capturing spatiotemporal features by enlarging the receptive field and addressing concerns related to gradient dispersion and explosion. In addition,

the self-attention mechanism in Informer resolves an issue of gradient vanishing and effectively captures longer time span information. Additionally, an activation function is employed to initialize the weight parameters. This serves to expedite the convergence speed of SCIS and bolster the sparsity of the extracted features. Finally, we adopt an improved metaheuristic optimization algorithm named GSPSO to fine-tune some key hyperparameters to enhance the accuracy of classification. Simulation results with a real-life data set in the Yahoo Webscope S5 prove that SCIS yields the best results in comparison with state-of-the-art benchmark methods.

Our future work plans to deepen our research in the following aspects. First, we plan to implement our proposed model with other data sets to demonstrate its generalizability. Second, we will improve the current model framework by incorporating more advanced deep learning models. Third, we will consider adding batch standardization and gradient clipping to optimize model training and reduce overfitting. Fourth, currently, we only focus on optimizing a limited number of hyperparameters, and we will optimize more network hyperparameters in the future.

REFERENCES

[1] A. Tsiota, D. Xenakis, N. Passas, and L. Merakos, "On jamming and black hole attacks in heterogeneous wireless networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10761–10774, Nov. 2019.

[2] J. Bi, K. Xu, H. Yuan, J. Zhang, and M. Zhou, "Network attack prediction with hybrid temporal convolutional network and bi-directional GRU," *IEEE Internet Things J.*, early access, Nov. 21, 2023, doi: 10.1109/JIOT.2023.3334912.

[3] R. Hwang, M. Peng, C. Huang, P. Lin, and V. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

[4] Y. Bi, G. Han, C. Lin, Y. Peng, H. Pu, and Y. Jia, "Intelligent quality of service aware traffic forwarding for software-defined networking/open shortest path first hybrid Industrial Internet," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1395–1405, Feb. 2020.

[5] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, "Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection," *Big Data Min. Analytics*, vol. 4, no. 1, pp. 18–24, Mar. 2021.

[6] W. Duo, M. Zhou, and A. Abusorrah, "A survey of cyber attacks on cyber physical systems: Recent advances and challenges," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 5, pp. 784–800, May 2022.

[7] M. Bhanu, J. Mendes-Moreira, and J. Chandra, "Embedding traffic network characteristics using tensor for improved traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3359–3371, Jun. 2021.

[8] J. Bi, Z. Guan, H. Yuan, and J. Zhang, "Improved network intrusion classification with attention-assisted bidirectional LSTM and optimized sparse contractive autoencoders," *Expert Syst. Appl.*, vol. 244, Jun. 2024, Art. no. 122966.

[9] A. G. Roselin, P. Nanda, S. Nepal, and X. He, "Intelligent anomaly detection for large network traffic with optimized deep clustering (ODC) algorithm," *IEEE Access*, vol. 9, pp. 47243–47251, 2021.

[10] N. Liu, W. Dai, R. Santerre, J. Hu, Q. Shi, and C. Yang, "High spatio-temporal resolution deformation time series with the fusion of InSAR and GNSS data using spatio-temporal random effect model," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 1, pp. 364–380, Jan. 2019.

[11] J. Bi, H. Yuan, J. Zhang, and M. Zhou, "Green energy forecast-based bi-objective scheduling of tasks across distributed clouds," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 3, pp. 619–630, Sep. 2022.

[12] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 9, pp. 5506–5517, Sep. 2022.

[13] L. Zhao, H. Yuan, K. Xu, J. Bi, and B. Li, "Hybrid network attack prediction with Savitzky–Golay filter-assisted informer," *Expert Syst. Appl.*, vol. 235, Jan. 2024, Art. no. 121126.

[14] J. Bi, H. Yuan, K. Xu, H. Ma, and M. Zhou, "Large-scale network traffic prediction with LSTM and temporal convolutional networks," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 3865–3870.

[15] R. Xin, P. Chen, and Z. Zhao, "CausalRCA: Causal inference based precise fine-grained root cause localization for microservice applications," *J. Syst. Softw.*, vol. 203, Sep. 2023, Art. no. 111724.

[16] X. Liu, "An abnormal network traffic detection method on MAWILab dataset based on convolutional neural network," in *Proc. IEEE 2nd Int. Conf. Electron. Technol., Commun. Inf.*, 2022, pp. 1233–1235.

[17] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate prediction of workloads and resources with multi-head attention and hybrid LSTM for cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375–384, Sep. 2023.

[18] K. Yoshihara and K. Takahashi, "A simple method for unsupervised anomaly detection: An application to Web time series data," *PLoS ONE*, vol. 17, no. 1, Jan. 2022, Art. no. e0262463.

[19] T. Koji and T. Kanji, "Dark reciprocal-rank: Teacher-to-student knowledge transfer from self-localization model to graph-convolutional neural network," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 1846–1853.

[20] Y. Jung, M. Jeon, C. Kim, S. W. Seo, and S. W. Kim, "Uncertainty-aware fast curb detection using convolutional networks in point clouds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 12882–12888.

[21] G. Yan, A. Schmitz, S. Funabashi, S. Somlor, T. P. Tomo, and S. Sugano, "SCT-CNN: A spatio-channel-temporal attention CNN for grasp stability prediction," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 2627–2634.

[22] H. Wang et al., "Risk assessment and mitigation in local path planning for autonomous vehicles with LSTM based predictive model," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 2738–2749, Oct. 2022.

[23] J. Bi, X. Zhang, H. Yuan, J. Zhang, and M. Zhou, "A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1869–1879, Jul. 2022.

[24] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. 23rd Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2015, pp. 22–24.

[25] M. Cheng, Q. Xu, J. Lv, W. Liu, Q. Li, and J. Wang, "MS-LSTM: A multi-scale LSTM model for BGP anomaly detection," in *Proc. IEEE 24th Int. Conf. Netw. Protocols*, 2016, pp. 1–6.

[26] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 4580–4584.

[27] J. A. Vincent and M. Schwager, "Reachable polyhedral marching (RPM): A safety verification algorithm for robotic systems with deep neural network components," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 9029–9035.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[29] J. Liu et al., "A graph attention spatio-temporal convolutional network for 3-D human pose estimation in video," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 3374–3380.

[30] T. Kim and S. Cho, "Web traffic anomaly detection using C-LSTM neural networks," *Expert Syst. Appl.*, vol. 106, pp. 66–76, Sep. 2018.

[31] L. Fu, W. Zhang, X. Tan, and H. Zhu, "An algorithm for detection of traffic attribute exceptions based on cluster algorithm in Industrial Internet of Things," *IEEE Access*, vol. 9, pp. 53370–53378, 2021.

[32] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Security Inform.*, 2017, pp. 43–48.

[33] Y. Ren and Y. Wu, "Convolutional deep belief networks for feature extraction of EEG signal," in *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 2850–2853.

[34] Y. Zhou and J. Li, "Research of network traffic anomaly detection model based on multilevel autoregression," in *Proc. IEEE 7th Int. Conf. Comput. Sci. Netw. Technol.*, 2019, pp. 380–384.

[35] M. Shajari, H. Geng, K. Hu, and A. Leon-Garcia, "Tensor-based online network anomaly detection and diagnosis," *IEEE Access*, vol. 10, pp. 85792–85817, 2022.

[36] T. M. Tran, T. N. Vu, T. V. Nguyen, and K. Nguyen, "UIT-ADrone: A novel drone dataset for traffic anomaly detection," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 16, pp. 5590–5601, Jun. 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10158513

[37] Y. Yao et al., "DoTA: Unsupervised detection of traffic anomaly in driving videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 444–459, Jan. 2023.
[38] H. Yuan, S. Wang, J. Bi, and J. Zhang, "Deep and spatio-temporal detection for abnormal traffic in cloud data centers," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2023, pp. 4985–4990.
[39] K. Kumaran Santhosh, D. P. Dogra, P. P. Roy, and A. Mitra, "Vehicular trajectory classification and traffic anomaly detection in videos using a hybrid CNN-VAE architecture," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11891–11902, Aug. 2022.
[40] L. Deng, D. Lian, Z. Huang, and E. Chen, "Graph convolutional adversarial networks for spatiotemporal anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2416–2428, Jun. 2022.
[41] Z. Zheng et al., "Anomaly detection of metro station tracks based on sequential updatable anomaly detection framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7677–7691, Nov. 2022.
[42] T. Phan, T. Nguyen, N. Dao, T. Huong, N. Thanh, and T. Bauschert, "DeepGuard: Efficient anomaly detection in SDN with fine-grained traffic flow monitoring," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1349–1362, Sep. 2020.
[43] X. Li et al., "Neighbor graph based tensor recovery for accurate Internet anomaly detection," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 2, pp. 655–674, Feb. 2023.
[44] J. Fan, K. Wu, Y. Zhou, Z. Zhao, and S. Huang, "Fast model update for IoT traffic anomaly detection with machine unlearning," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8590–8602, May 2023.
[45] W. Huang, K. Xie, and J. Li, "A novel sequence tensor recovery algorithm for quick and accurate anomaly detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3531–3545, Jul. 2022.
[46] H. Yuan, Q. Hu, and J. Bi, "Cost-minimized and multi-plant scheduling in distributed industrial systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2022, pp. 2851–2856.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020.

He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

Shen Wang (Graduate Student Member, IEEE) received the B.S. degree in quality and reliability of aircraft from Beihang University, Beijing, China, in 2022, where he is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering.

His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Mr. Wang was the recipient of the Merit Student of Beihang University in 2019.

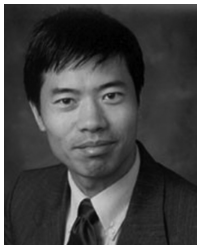
Jing Bi (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively.

From 2013 to 2015, she was a Postdoctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China. From 2011 to 2013, she was a Research Scientist with Beijing Research Institute of Electronic Engineering Technology, Beijing. From 2009 to 2010, she was a Research Assistant and participated in research on cloud computing with IBM Research, Beijing. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud computing, large-scale data analytics, machine learning, and performance optimization.

Dr. Bi received the IBM Fellowship Award, the Best Paper Award at the 17th IEEE International Conference on Networking, Sensing and Control, and the First-Prize Progress Award of the Chinese Institute of Simulation Science and Technology. She is currently an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, the Professor of the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science.



MengChu Zhou (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990

He was then with New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. His interests are in Petri nets, automation, Internet of Things, cloud/edge computing, and AI. He has more than 1000 publications, including 12 books, more than 700 journal papers (more than 600 in IEEE transactions), 31 patents, and 30 book chapters.

Dr. Zhou is a Fellow of IFAC, AAAS, CAA, and NAI.