Network Attack Prediction With Hybrid Temporal Convolutional Network and Bidirectional GRU

Jing Bi, *Senior Member, IEEE*, Kangyuan Xu, *Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Precise and real-time prediction of future network attacks can not only prompt cloud infrastructures to fast respond and protect network security but also prevents economic and business losses. In recent years, neural networks, e.g., bidirectional gated recurrent unit (Bi-GRU) network and temporal convolutional network (TCN), have been proven to be suitable for predicting time-series data. Attention mechanisms are also widely used for the prediction of the time series of network attacks. This work proposes a hybrid deep learning prediction method that combines the capabilities of Savitzky–Golay (SG) filter, TCN, multihead self-attention, and Bi-GRU (STMB) for the prediction of network attacks. This work first adopts an SG filter to smooth possible outliers and noise in network attack traffic data. It applies TCN to extract abstract features from 1-D time series to make full use of data. It then adopts multihead self-attention to capture internal correlations among multidimensional features, by increasing the weights of key features and reducing those weight of non-key features, making that STMB captures important features adaptively. Finally, this work adopts Bi-GRU to extract bidirectional and long-term correlations in the time series to improve the prediction accuracy. This work also utilizes a hybrid algorithm named genetic simulated-annealing-based particle swarm optimizer to determine the hyperparameter setting of STMB. Experimental results with real-life data sets show that STMB outperforms several commonly used algorithms in terms of prediction accuracy.**

***Index Terms*—Gated recurrent unit (GRU), multihead self-attention, network attack prediction, Savitzky–Golay (SG) filter, temporal convolutional network (TCN).**

Manuscript received 21 December 2022; revised 20 February 2023, 17 April 2023, 5 July 2023, and 9 November 2023; accepted 15 November 2023. Date of publication 21 November 2023; date of current version 26 March 2024. This work was supported in part by the Beijing Natural Science Foundation under Grant 4232049 and Grant L233005; in part by the National Natural Science Foundation of China (NSFC) under Grant 62173013 and Grant 62073005; 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 IEEE International Conference on Systems, Man and Cybernetics (SMC 2022), Prague, Czech Republic, 9–12 October 2022 [DOI: 10.1109/SMC53654.2022.9945189]. *(Corresponding author: Haitao Yuan.)*

Jing Bi and Kangyuan Xu are with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn; kyxu1999@emails.bjut.edu.cn).

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

Jia Zhang is with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX 75205 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.2023.3334912

I. Introduction

NETWORK attacks are offensive actions against computer information systems, infrastructure, computer networks, or personal computer devices [1]. As the use of cloud services increases, a growing number of users of Web applications lead to changes in the network infrastructure that connects devices running on mobile operating systems, enabling network technologies to evolve [2], [3]. Current cloud infrastructures face a growing number of network attacks with more and more types every day, which bring big challenges to the network security [4]. Therefore, network attacks threaten the security of countries and global users all the time. It is important to pay attention to the network attacks. Accurately predicting the number of future network attacks is an effective way to provide preventive measures for the network security in advance.

Predicting network attack traffic for a future period falls into a category of time-series prediction. Time-series prediction methods can be divided into two categories, including traditional methods and deep learning ones. Traditional linear methods, such as autoregressive (AR) [5], AR moving average (ARMA), and AR integrated MA (ARIMA) [6], are widely adopted in the time-series data prediction. Calheiros et al. [7] realized a cloud workload prediction module based on ARIMA. However, ARIMA only captures linear relations in the data, but fails to investigate nonlinear ones. To capture nonlinear features in the time series, researchers turn their attentions to models suitable for complex and nonlinear data. Among them, support vector machine (SVM) is a widely adopted. However, storage and calculation of matrices consume huge memories and computing time when dealing with large-scale data.

The data set of network attack activities has characteristics of huge volume, large fluctuation, nonlinear changes, etc., which limits the effect of traditional machine learning algorithms. First, the size of the data set affects the speed and accuracy of prediction methods. Traditional methods cannot handle a large amount of data, resulting in lower prediction accuracy. Second, time-series data with large fluctuations require multiple time slots for analysis. Traditional methods mainly adopt a small number of time slots for analysis, and therefore, it is difficult to capture key features of the data set. Finally, traditional methods often adopt linear models and fail to capture complex nonlinear relations in the data set of network attacks, thereby yielding larger prediction errors. Compared with traditional methods,

deep learning methods have significant advantages in network attack prediction. First, deep learning methods can better capture the nonlinear relationship in the network attack data. Second, they have the capability of large-scale data processing and analysis. Third, they can be better generalized to analyze the new data, which means they are more stable in the network attack prediction over different time periods or in different network environments. Therefore, a deep learning-based method is highly needed to predict future network attacks. Recurrent neural networks (RNNs) are widely used to predict the time-series data by mining temporal information in the data. Long short-term memory (LSTM) is one of the variants of RNNs, and it can alleviate gradient disappearance problems. As another variant of RNNs, gated recurrent units (GRUs) are suitable for building larger networks and have only two gates, thus providing efficient computing. Despite the popularity of RNNs, convolutional neural networks (CNNs) have better performance and accuracy than RNNs in some cases. In recent years, temporal convolutional network (TCN), which combines characteristics of RNNs and CNNs, has become an important method for the time-series prediction. Wang et al. [8] adopted TCN for short-term and composite forecasting of industrial users. TCN extracts the time relationship between historical time series and features in a long-time range, thereby yielding better performance. Models based on transformer [9] also show better advantages in time-series data. It adopts a self-attention mechanism to analyze the time-series data, from which complex dependencies of different lengths can be learned.

This work proposes a hybrid method named STMB based on a Savitzky–Golay (SG) filter, a TCN, a multihead self-attention, and a bidirectional GRU (Bi-GRU) network. To integrate these components, the best model combination of TCN, multihead self-attention, and Bi-GRU is determined. First, the data filtered by the SG filter is 1-D time-series data, and it cannot be directly processed by the multihead self-attention. Thus, feature decomposition is required to ensure that the data dimension in the multihead self-attention is divisible by the number of heads. Second, the convolution operation in TCN is suitable to extract abstract features in the 1-D time series because it decomposes 1-D data into multidimensional data. Third, the dimensional input of the multihead self-attention is met by TCN, and the multihead self-attention captures internal correlations among multiple features. Then, the weights of key features are increased while those of non-key features are reduced. Finally, Bi-GRU is used to capture bidirectional, long-term, and short-term dependence among different features to realize the prediction. In addition to this, we adopt genetic simulated annealing-based particle swarm optimization (GSPSO) [10] to optimize the hyperparameter settings of STMB. Experimental results show that STMB is superior to the most advanced baseline models on prediction accuracy. The main contributions of this work can be summarized into two aspects.

1) A raw sequence is logarithmically processed to approximate a normal distribution, and further used to eliminate strong noise with an SG filter. After the data preprocessing stage, short-term local features are first extracted using TCN, then intrinsic connections among features are captured using multihead self-attention, and finally, the bidirectional and long-term correlations in the sequence are extracted by using Bi-GRU.
2) Combining TCN and multihead self-attention with Bi-GRU, an excellent hybrid prediction model, STMB, is designed for network attack prediction. It is characterized by extracting abstract features from 1-D time series, assigning different weights to different features, and capturing bidirectional, long and short-term dependencies between different features. STMB provides a new mechanism that integrates noise removal, capturing internal correlations of abstract features, bidirectional and long-term ones of network attack time series, thereby yielding higher prediction accuracy. In addition, this work designs a novel algorithm named GSPSO to optimize the hyperparameter settings of STMB. Through the experiments on two real-life data sets, STMB is proven to consistently outperform its updated peers.

For clarity, we note major differences between the current one and our prior work [11] as follows.

1) The study in [11] adopts a bidirectional LSTM (Bi-LSTM) model to extract bidirectional and long-term correlations in the sequences. Bi-GRU has no gated forgetting units and owns fewer parameter updates during the training, making it less computationally intensive and faster to train. Thus, this work adopts Bi-GRU to improve the convergence speed and prediction accuracy of STMB.
2) The study in [11] determines the setting of the number of previous time steps only by a limited number of trials. GSPSO automatically determines the hyperparameter setting of STMB according to the characteristics of the data, which avoids the low prediction accuracy caused by manually fixed parameters. Thus, this work adopts GSPSO to optimize it for finding the best setting in its given range, thereby yielding higher prediction accuracy.
3) The study in [11] only adopts a single type of data set. Different from it, this work adopts two types of different data sets to demonstrate the prediction performance and robustness of our proposed STMB. Two types of different data sets have different periodic changes, noises, and outliers. The experimental results demonstrate that STMB yields better prediction results than several state-of-the-art prediction models given two data sets.

The remainder of the work is organized as follows. We describe the related work in Section II and show the proposed method in Section III. The experimental results and discussions are presented in Section IV. Finally, conclusions are drawn in Section V.

II. Related Work

Accurate and real-time prediction of network attacks significantly reduces the loss of network facilities, and effective actions can be taken. Current studies about the prediction of network attacks can be mainly divided into classical network attack methods and deep learning-based network attack ones.

A. Classical Network Attack Methods

Classical network attack methods aim to predict and detect different network attacks separately with machine learning methods like Markov stochastic processes and Bayesian networks to obtain the intent of network attackers. Moudoud et al. [12] proposed a Markov stochastic process-based security model for predicting and detecting network attacks. Due to the limitation of Markov randomness, it is only suitable for short-term prediction and its effect of medium and long-term prediction is not good. Huang et al. [13] involved Bayesian networks for attack prediction in a framework. However, there are no improvement in the prediction method itself and it cannot make long-term prediction. Data from honeypots are used by Bar et al. [14] for modeling of attack propagation with Markov chains, and they observed several frequent patterns of attack propagation. However, it cannot predict the scale of future network attacks.

In addition to the above prediction methods, there is another class of methods to predict network attacks by using the time-series data with machine learning methods like AR and ARIMA. Werner et al. [15] adopted ARIMA to predict the number of network attacks in the coming day. However, its data is not preprocessed and ARIMA only captures linear relations in the data, but fails to investigate nonlinear ones. Okutan et al. [16] designed a network attack prediction system. It adopts a set of signals predicted by ARIMA to improve the prediction performance. However, ARIMA cannot capture nonlinear features of the sequence only by the approximate linear fitting. Bi et al. [17] proposed a method that combines wavelet decomposition and ARIMA to predict network tasks. However, ARIMA is highly sensitive to local outliers, and the prediction results are significantly affected.

Different from them, this work predicts the number of network attacks in the future by using time-series data with deep learning methods to prepare for dealing with network attacks in advance. Deep learning methods capture nonlinear features in the sequence and achieve higher predict performance.

B. Deep Learning-Based Network Attack Methods

Due to improved computing power and the emergence of optimization algorithms, deep learning methods have become popular in predicting and detecting network attacks. Lu et al. [18] proposed a deep belief network detection method based on total extreme value optimization to detect network attacks. However, its training requires a large number of data samples, and the process is more complicated, requiring a long time of training and optimization. Al-Abassi et al. [19] proposed an attack detection model that adopts a deep neural network and a decision tree classifier to detect cyber-attacks from new representations. However, this method is trained on a single data set, and it is difficult to extend to other network environments. Ganesh et al. [20] designed a layered long and short-term memory model to process raw data streams from relevant online cyber–physical system sensors and continuously monitor embedded signals in the data to detect and characterize attacks. However, the modeling ability of this method for long sequences is weak. Current deep learning-based methods are mainly about detecting and identifying incoming network attacks. In the face of such increasingly intelligent, complex, and massive network attacks, traditional methods cannot accurately predict malicious intrusion network attacks. Thus, it is equally important to predict the number of network attacks in the future moment.

Different from these studies, we innovatively combine TCN and Bi-GRU and add a multihead self-attention mechanism into the model to further improve the prediction accuracy of future network attacks based on past network attack traffic. Specifically, after the processing by the SG filter, TCN extracts short-term and local features in the sequence, and the multihead self-attention mechanism captures intrinsic correlations of features. Finally, Bi-GRU captures bidirectional and long-term correlations in the sequence to realize the final prediction.

TABLE I
LIST OF ABBREVIATIONS

Abbreviation	Definition
MA	Moving Average filter
MM	Median filter
SG	Savitzky-Golay filter
PSO	Particle Swarm Optimization
GSPSO	Genetic Simulated annealing-based PSO
INDICS	INDustrial Intelligent Cloud System
AR	AutoRegressive
ARMA	AutoRegressive Moving Average
ARIMA	AutoRegressive Integrated Moving Average
SVM	Support Vector Machine
SVR	Support Vector Regression
BPNN	Back Propagation Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM	Bi-directional Long Short-Term Memory
GRU	Gated Recurrent Unit
Bi-GRU	Bi-directional Gated Recurrent Unit
CNN	Convolutional Neural Networks
TCN	Temporal Convolutional Network
T-LSTM	TCN+LSTM
T-BiLSTM	TCN+Bi-LSTM
TMB	TCN+Multi-head self-attention+Bi-GRU
ST-LSTM	SG filter+TCN+LSTM
ST-BiLSTM	SG filter+TCN+Bi-LSTM
STMB	SG filter+TCN+Multi-head self-attention+Bi-GRU

III. PROPOSED METHODOLOGY

This section introduces the details of STMB. To fully exploit their advantages, we combine four parts into a novel hybrid model to further improve the prediction accuracy. First, the sequence problem is described in Section III-A. Second, we introduce the details of the SG filter, TCN, the multihead self-attention, and Bi-GRU, respectively, in Sections III-B–III-E. And the details of the overall framework of STMB are given in Section III-F. Third, we present the GSPSO used in the experiment to optimize the hyperparameter setting and its specific steps. Finally, we introduce our training procedure in Section III-H. For clarity, a list of abbreviations of our method is summarized in Table I.

A. Problem Definition

Sequence modeling in the time-series forecasting has been widely used. $X = \{x_1, \ldots, x_t, \ldots, x_T\}$ denotes the time series and the length of the time series is T. x_t denotes the number of network attacks in time slot t. $\bar{X} = \{\bar{x}_1, \ldots, \bar{x}_t, \ldots, \bar{x}_T\}$ denotes a sequence with the length of T, processed by the SG filter. $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_t, \ldots, \hat{x}_T\}$ denotes a sequence with the length of T after the feature extraction phase with TCN. $\widetilde{X} = \{\widetilde{x}_1, \ldots, \widetilde{x}_t, \ldots, \widetilde{x}_T\}$ denotes a sequence with the length of T after the feature selection phase with the multihead self-attention. $\hat{y}_T$ and y_T denote the predicted value with the step length of 1 and its ground-truth value, respectively. The data in previous T time steps is used to predict the value at the time step $T+1$. Our goal is to minimize the prediction error to acquire the nonlinear mapping from the input to the predicted result. Thus, a nonlinear function of $\mathrm{SM}(\cdot)$ is defined as

$$\hat{y}_T = \mathrm{SM}(X). \tag{1}$$

B. Savitzky–Golay Filter

Time-series data can be predicted more accurately by smoothing and de-noising the original sequence [21]. We adopt the SG filter [22] to achieve this because it ensures that the shape and width of the data remain constant while filtering noise. A subsequence of X with a window size of $n = 2m+1$ is expressed as

$$\{x_{s-m}, \ldots, x_s, \ldots, x_{s+m}\}, \quad s \in [m+1, T-m]. \tag{2}$$

The R-order polynomial $\mathrm{p}(i)$ adopted to fit data points within the window is defined as

$$p(i) = \sum_{v=0}^{R} a_v i^v, \quad i \in [-m, m] \tag{3}$$

where a_v denotes the vth coefficient of the SG filter.

Then, we adopt the least-square method to minimize the following error ϵ, which is defined as:

$$\epsilon = \sum_{i=-m}^{m} (\mathrm{p}(i) - x_{s+i})^2. \tag{4}$$

Then, the best fitting $p(0)$ of the window center point x_s is obtained by calculating a_0. By translating the window, each point in X is a center point in the window. Finally, the smoothed sequence $\bar{X}$ is obtained.

C. Temporal Convolutional Network

TCN is a special 1-D fully CNN [23], which includes causal convolution, dilated one, and residual block. Causal convolution ensures that the value in time step t in the upper layer depends only on those in time step t and before in the lower layer. For a one-dimension input l and a filter $f : \{0, 1, \ldots, k-1\}$, the 1-D casual convolutional layer is expressed as

$$F(l_t) = (l \star f)(t) = \sum_{j=0}^{k-1} f_j l_{t-j} \tag{5}$$

$$\text{sequence} = (F(l_1), F(l_2), \ldots, F(l_T)) \tag{6}$$

where *sequence* denotes an output sequence, $F(\cdot)$ denotes a convolutional operation, and k denotes the convolutional kernel size.

The dilated convolution skips a part of the input with hyperparameters. Thus, the filter can operate at a range larger than itself. When it is combined with the causal convolution, the dilated convolution of the rth layer is defined as

$$F(l_t) = \left(l \star_{d_r} f\right)(t) = \sum_{j=0}^{k-1} f_j l_{t-d_r j} \tag{7}$$

$$\text{sequence} = (F(l_1), F(l_2), \ldots, F(l_T)) \tag{8}$$

where d_r denotes a dilation factor of layer r, which can be set to 2^{r-1}. More details about (7) can be found in previous studies [11]. Equation (8) represents a temporal convolutional layer, and TCN is constructed by stacking multiple layers.

The temporal convolutional layers are combined into blocks and residual connections are placed among the blocks. Each residual block has two dilated convolution layers, which have a rectified linear units (ReLUs) function. In addition, TCN realizes the regularization by adding dropout [24] to each residual block after the dilated convolution.

D. Multihead Self-Attention

This work adopts the multihead self-attention [25] as an attention module. The input is the sequence $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_t, \ldots, \hat{x}_T\}$ after TCN's feature extraction. Inside the module, there are three matrices, query (Q), key (K), and value (V). Q, K, and V are linearly transformed from the input matrix. First, they are projected linearly onto Q', K', and V'. Furthermore, they are divided into multiple heads to extract the related information from different features. Each head in Q, K, and V is denoted by Q'_h, K'_h, and V'_h, where $h \in \{1, 2, \ldots, H\}$ and H denotes the number of heads. In query Q'_h, the weight distribution over the entire sequence is calculated based on the similarity between the query (Q'_h) and the key (K'_h). For each query, the compact and dynamic representation brings more related information into the sequence by adding weights to the value (V'_h). We adopt a scaled dot operation to measure the similarity, which is given as

$$\mathrm{Attention}\left(Q'_h, K'_h, V'_h\right) = \mathrm{SoftMax}\left(\frac{Q'_h K'_h}{\sqrt{d_{K'_h}}}\right) V'_h \tag{9}$$

where $\mathrm{SoftMax}(\cdot)$ denotes a softmax function, and $d_{K'_h}$ denotes the dimension of a vector in K'_h.

Then, we combine multiple attentions by concatenating these head vectors. Finally, the combined attention enters a linear layer to obtain new representations. Here, $Q = K = V$

$$\mathrm{head}_h = \mathrm{Attention}\left(Q'_h, K'_h, V'_h\right) \tag{10}$$

$$\mathrm{Multihead} = \mathrm{Concat}(head_1, \ldots, head_h)W. \tag{11}$$

E. Bi-GRU

The yielded result after the multihead self-attention is $\widetilde{X} = \{\widetilde{x}_1, \ldots, \widetilde{x}_t, \ldots, \widetilde{x}_T\}$. $\widetilde{X}$ is adopted as the input of the Bi-GRU to learn the mapping from $\widetilde{x}_t$ to h_t at time step t. h_{t-1}

and h_t denote the hidden states of Bi-GRU at time steps $t-1$ and t, respectively. h_t is obtained as

$$h_t = f(h_{t-1}, \widetilde{x}_t). \tag{12}$$

The nonlinear function $f(\cdot)$ is a GRU. To model long and short-term behaviors, Bi-GRU includes a forward GRU network and a backward one [26], thereby properly keeping and forgetting the past information. The cell unit structure of a GRU network consists of two gates, i.e., a reset gate and an update one. The reset gate decides how much information from a previous state to forget. The update gate is used to control how much information from the previous state (from previous time steps) needs to be passed to the current state and the future, and it decides to copy the past information and eliminate a risk of the problem of vanishing gradient. The calculation formulas of GRU from the input to the output are obtained as

$$r_t = \sigma\left(\boldsymbol{W_r} \cdot \left[h_{t-1}, \widetilde{x}_t\right]\right) \tag{13}$$

$$z_t = \sigma\left(\boldsymbol{W_z} \cdot \left[h_{t-1}, \widetilde{x}_t\right]\right) \tag{14}$$

$$\tilde{h}_t = \tanh\left(\boldsymbol{W}_{\tilde{\boldsymbol{h}}_t} \cdot \left[r_t * h_{t-1}, \widetilde{x}_t\right]\right) \tag{15}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{16}$$

$$y_t = \sigma(\boldsymbol{W_o} \cdot h_t) \tag{17}$$

where r, z, h_t, and $\tilde{h}$ denote a reset gate, an update one, the hidden state at time step t, and a candidate activation vector, respectively. $\boldsymbol{W_r}$, $\boldsymbol{W_z}$, and $\boldsymbol{W}_{\tilde{\boldsymbol{h}}_t}$ denote their weight matrices. $\sigma(\cdot)$ and $\tanh(\cdot)$ denote sigmoid and hyperbolic tangent functions, respectively. $*$ denotes the operation of matrix multiplication.

F. STMB

Our proposed STMB integrates four components, including the SG filter, TCN, multihead self-attention, and Bi-GRU. First, the 1-D time-series data is first filtered by the SG filter. Second, the 1-D data is decomposed by TCN to yield multidimensional data, thus extracting abstract features in the sequence. Third, the multihead self-attention component captures internal correlations among multiple abstract features. In this way, the weights of key features are increased, and ones of non-key features are reduced. Fourth, Bi-GRU is used to capture the bidirectional, long-term, and short-term dependence among different features to realize the prediction. Fig. 1 illustrates a framework of STMB.

First, the SG filter is adopted to denoise X and yield the smoothed result $\bar{X}$, which is given as

$$\bar{X} = \mathrm{SG}(X, n, R) \tag{18}$$

where $\mathrm{SG}(\cdot)$ denotes a function for the SG filter, n denotes its window size, and R denotes its polynomial order.

Second, $\bar{X}$ is extracted by TCN that has two residual blocks. The first block is composed of two causal dilated convolution layers. The kernel size is set to 9, the dilation factor is set to 1, and the number of filters is set to 10. In the second block, the kernel size is set to 9, the dilation factor is set to 2, and the filter number is set to 10. The output of the TCN layer is given as

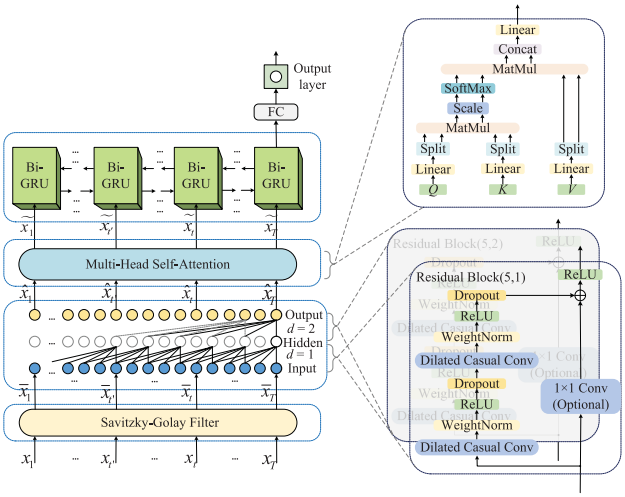


Fig. 1. Framework of our proposed STMB.

$$L = \mathrm{ResB}\left(\bar{X}, 9\right) \tag{19}$$

where $\mathrm{ResB}(\cdot)$ denotes a residual block function, and L denotes the output after the residual block.

Third, we take $\tilde{L}$ as the input to the multihead self-attention mechanism to identify useful features. Q, K, and V are obtained from the input. The output of the attention is

$$\widetilde{X} = \mathrm{Multihead}(L, n_h) \tag{20}$$

where n_h denotes the number of heads.

Fourth, this work takes $\widetilde{X}$ as the input to the Bi-GRU layer to extract bidirectional and long-term correlations in the sequence and acquire the output h_T at time step T. h_T is inputted into a fully connected layer to acquire the output z_T, which is defined as

$$z_T = \mathrm{ReLU}(vh_T + b) \tag{21}$$

where $\mathrm{ReLU}(\cdot)$, v, and b denote a ReLU activation function, a weight matrix, and a bias vector, respectively.

Finally, z_T is inputted into an output layer to acquire $\hat{y}_T$ at the next time step

$$\hat{y}_T = \mathrm{Linear}(uz_T + q) \tag{22}$$

where $\mathrm{Linear}(\cdot)$ denotes a linear function, u denotes a weight parameter of a fully connected layer, and q is a bias parameter.

G. GSPSO

PSO [27] has fast convergence and simple implementation, which is suitable for real-valued processing. However, it fails to handle well discrete optimization problems and tends to fall into local optima. In addition, the Metropolis acceptance rule in simulated annealing [28] allows accepting certain deteriorated solutions, thereby enabling to search more solutions in the solution space. However, its convergence speed is very slow. In addition, genetic operations in the genetic algorithm [29] can automatically acquire and accumulate

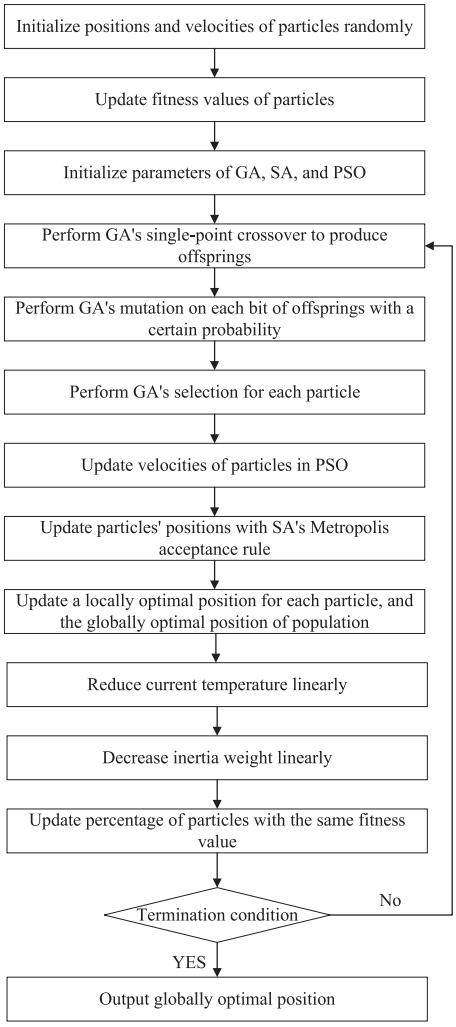


Fig. 2. Main steps of GSPSO.

knowledge about the search space, and adaptively control the search process to obtain the best solution, thereby improving the accuracy and efficiency of global search. The motivation of combining PSO and GA is given as follows. Genetic operators in GA can generate exemplars from which particles in PSO learn. In addition, the historical search information of particles in PSO provides guidance to the evolution of the exemplars. By realizing crossover, mutation, and selection on the historical information of particles in PSO, yielded exemplars are highly qualified and well diversified. Under such guidance, GSPSO's global search ability and efficiency are both improved. Thus, GSPSO integrates the advantages of the above three basic algorithms, providing a high convergence speed and strong global search capability. Fig. 2 shows the main steps of GSPSO.

Specifically, the position and velocity of each particle are initialized randomly, and their fitness values are updated. Then, the parameters of GA, SA, and PSO are initialized, and GA performs a single-point crossover to produce offspring. Afterward, GA performs mutation on each bit of each offspring with a certain probability and performs selection on each particle. Then, GSPSO updates the velocity of each particle in PSO and updates their position with SA's Metropolis acceptance criterion. In addition, GSPSO updates the locally best solution for each particle as well as the globally best solution in the current population. Furthermore, the current temperature and inertia weight are linearly decreased. Finally, the percentage of particles with the same fitness value is changed to determine if the termination criteria are met. If so, the globally best solution is obtained; otherwise, GA's single-point crossover and subsequent operations are repeated until the termination criteria are met. This work optimizes the hyperparameter setting of STMB with GSPSO. The loss value of STMB is used as the fitness value of each individual in GSPSO, and the hyperparameters of STMB are designed as decision variables in GSPSO. In this way, GSPSO determines the optimal hyperparameter setting that minimizes the training loss of the proposed STMB.

H. Training Procedure

To obtain better prediction accuracy, this work adopts the root-mean-square logarithmic error (RMSLE) [30], [31] as a loss function to minimize the difference between the ground-truth value y_T and the predicted one $\hat{y}_T$. The loss function is defined as

$$\text{loss} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log\left(\hat{y}_i+1\right)-\log(y_i+1)\right)^2}. \tag{23}$$

The rationale is that the magnitude of data may be relatively large. When there is a large difference between a small number of predicted values and the ground-truth ones in the data, the log function can reduce the influence of these predicted ones on the overall errors. During the training process of STMB, there are some crucial parameters for learning performance and efficiency. This work adopts a widely adopted Adam optimizer with fast convergence [32] to optimize the loss function of STMB.

IV. Experimental Evaluation

STMB is implemented on a computer with RTX3060 GPU, 16G memory, and Intel Core i7-11800H CPUs with 8-core 2.30-GHz processors.

A. Data Set Description

The network attack data is collected from 1 June 2021 to 30 August 2021 from two different cities in the Industrial Intelligent Cloud System (INDICS). Each dot in the x-axis represents a sample in each time slot, and each sample represents the number of attacks in the website within 10 min. In total, we have 13 200 samples in data set 1, and 12 644 samples in data set 2, and the ratio of the training set and the test one is 9:1. Fig. 3 shows the number of network attacks in each time slot in data set 1, and Fig. 4 shows that in data set 2.

B. Experimental Procedure

First, the SG filter is used to remove the noise in the original data. The max–min normalization is used to normalize the de-noised data. A sliding window mechanism is used to transform it into the supervised data, which is predicted by

TABLE II
DIFFERENT WINDOW SIZES OF RMSLE WITH DIFFERENT FILTERS

Window sizes	SG Filter		MA Filter		MM Filter	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
5	**0.0784**	**0.0886**	0.1461	0.3155	0.1148	0.3065
7	**0.0964**	**0.1154**	0.1492	0.3269	0.1182	0.3153
9	**0.1032**	**0.1265**	0.1573	0.3946	0.1208	0.3674
11	**0.1089**	**0.1286**	0.1554	0.3980	0.1234	0.3965

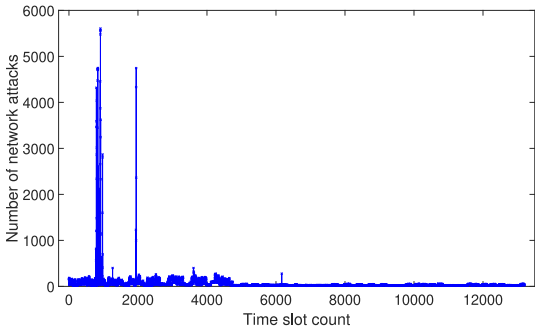


Fig. 3. Network attack time series in data set 1.

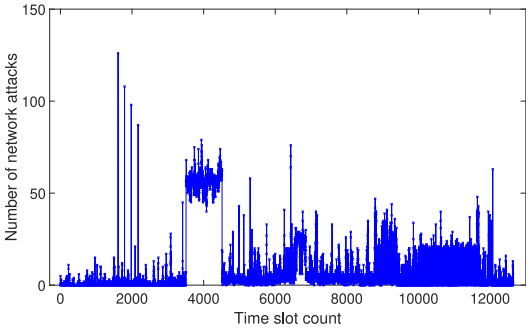


Fig. 4. Network attack time series in data set 2.

STMB. The prediction model integrated with TCN, multihead self-attention, and Bi-GRU is obtained after iterative training with super parameters. GSPSO is used to adjust and optimize super parameters to obtain the final STMB model. The final selection of super parameters in the model is shown in Sections IV-D and IV-E. Finally, the final model after training is used to predict the number of future network attacks, and the predicted value is reversely normalized, thus obtaining the predicted number of future network attacks. The predicted value is compared with its ground truth, and error analysis and performance evaluation are carried out.

C. Baseline Algorithms

We compare STMB with its typical peers, including ARIMA, SVR [33], BPNN [34], TCN, LSTM, Bi-LSTM [35], and TCN+LSTM (T-LSTM) [36]. We apply the SG filter to each baseline method to eliminate noise from the data, thus resulting in S-ARIMA, S-SVR, S-BPNN, S-TCN, S-LSTM, and ST-LSTM, respectively. Furthermore, we combine TCN, multihead self-attention, and Bi-GRU (TMB) to obtain an improved benchmark. Three metrics are adopted to evaluate the prediction accuracy of network attacks, i.e., RMSLE, a

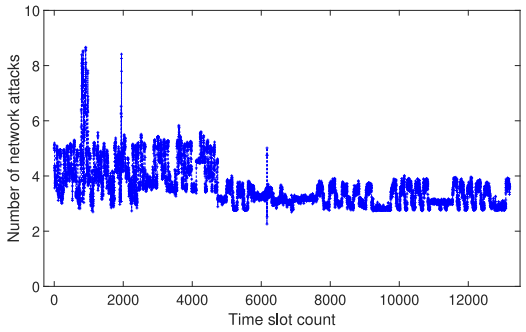


Fig. 5. Smoothed time series of data set 1 (log).

determination coefficient R^2, and the mean absolute error (MAE) [39], which are given as

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(\hat{y}_i+1)-\log(y_i+1)\right)^2} \tag{24}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i-\bar{y}\right)^2} \tag{25}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{26}$$

where $\hat{y}$ denotes the predicted value, y denotes the real one, and $\bar{y}$ denotes the average one of the real values.

D. Noise Filter Tuning

As shown in Figs. 3 and 4, the raw data has high-order and nonlinear characteristics. Thus, the logarithmic operation is adopted to make the data approximately obey the normal distribution. Fig. 3 shows that it has obvious noise and several peak points at the beginning, and Fig. 4 also shows that the raw data has obvious noise and several peak points. Thus, we need to choose a suitable filter to eliminate noise, which has two important hyperparameters, i.e., the window size (n) and the polynomial order (R). The median filter and the MA one are widely adopted and compared with the SG filter. The window size is selected from {5, 7, 9, 11}. From Table II, it is shown that among three filters, the SG filter achieves the best filtering performance. The reason is that the SG filter can keep the shape and width of the signal unchanged while eliminating noise. This demonstrates that the SG filter effectively eliminates the noise. Figs. 5 and 6 show the smoothed time series of data sets 1 and 2 with the SG filter.

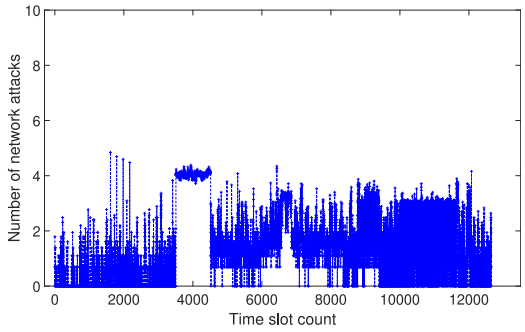


Fig. 6. Smoothed time series of data set 2 (log).

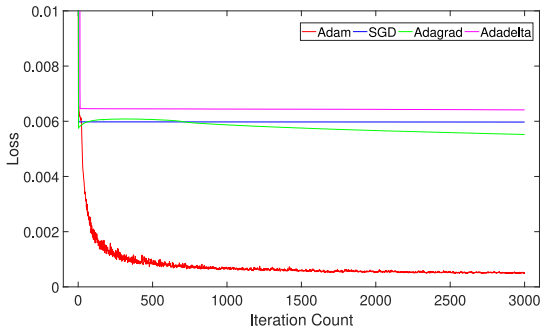


Fig. 7. Loss value for each optimizer.

E. Parameter Tuning

The setting of hyperparameters greatly affects the performance of STMB. Hyperparameters in STMB mainly include the optimizer, the kernel size, the number of attention heads, the number of neurons in Bi-GRU, and the output activation function of TCN residual blocks. This work also adopts GSPSO to optimize the size of previous time steps. Similar to previous studies [37], the parameter selection of GSPSO is realized by using a grid search method [38]. Specifically, the main parameters of GSPSO are set as follows. The coefficients of individual and social acceleration are both set to 0.5. The acceleration coefficient of each superior particle is 1.5. The starting temperature is 10^8. The cooling rate of temperature is 0.95. The upper limit of inertia weight is 0.95 and the lower one is 0.4. In addition, we add a limit to the size of previous time steps from 30 to 90, and finally set it to 60 according to GSPSO for yielding better prediction results.

The optimizer is very important, and we compare four candidates including Adaptive delta (Adadelta), Adaptive gradient algorithm (Adagrad), stochastic gradient descent (SGD), and Adaptive moment estimation (Adam). Fig. 7 shows that compared with other peers, Adam has the fastest convergence speed and the minimum loss value for data set 1. The SGD optimizer uses a small amount of data for each parameter update, resulting in a large oscillation amplitude during gradient update. The AdaGrad optimizer can automatically update different learning rates for different parameters. However, with the increase of time steps, the learning rate cannot be updated effectively. The AdaDelta optimizer in the early and middle training, the acceleration effect is good. However, in the late training period, it repeatedly shakes around the local minima. The Adam optimizer integrates the advantages of Adagrad and Momentum to alleviate the gradient shock problem. Thus, Adam is selected as the optimizer in STMB.

TABLE III
COMPARISON OF DIFFERENT KERNEL SIZES IN STMB

Kernel sizes	RMSLE		MAE		R^2	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
3	0.0829	0.1054	1.7581	1.9253	0.9365	0.8915
5	0.0844	0.1048	1.7283	1.9021	0.9405	0.8936
7	0.0815	0.0943	1.7241	1.8205	0.9401	0.9012
9	**0.0780**	**0.0886**	**1.6691**	**1.7457**	**0.9460**	**0.9154**
11	0.0809	0.0993	1.7173	1.8255	0.9413	0.9001
13	0.1666	0.1754	3.3605	3.5756	0.7541	0.7345

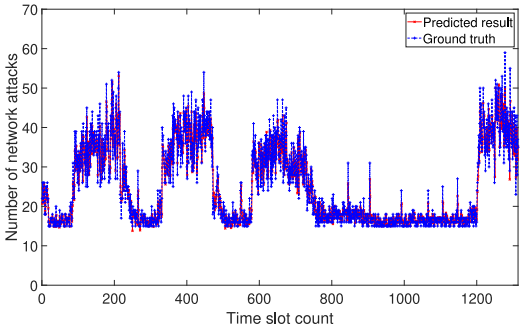


Fig. 8. Prediction results with STMB with data set 1.

A properly specified convolution kernel helps to learn more features in the data. Table III shows that when the convolutional kernel size is set to 9, R^2 is the largest, and RMSLE and MAE are the smallest. A larger kernel size leads to a larger receptive field of TCN. If the kernel size is too large, TCN degenerates into a full connection layer. Therefore, the kernel size cannot be too large or too small. Thus, it is set to 9. In the multihead self-attention mechanism, each head is an independent attention, and finally, all of them are integrated to prevent the overfitting. It is important to choose a suitable number of heads. If the number of heads is too large, it adds a huge amount of computation. If it is too small, it degenerates into a normal self-attention mechanism. Table IV shows that when it is set to 4, STMB achieves the best result. Thus, it is set to 4.

The suitable number of neurons in Bi-GRU has an important influence on the prediction accuracy. It is set to 32, 64, 72, and 128, respectively, and we choose the best number of neurons in Bi-GRU. Table V shows that RMSLE and MAE are the smallest, and R^2 is the largest, when it is set to 64. Regarding TCN, the ReLU activation function is adopted after each residual block. Table VI demonstrates the comparison of five activation functions including ReLU, LeakyReLU, Sigmod, Tanh, and Softplus. Experimental results show that among five activation functions, ReLU achieves the best result. The ReLU activation function can avoid gradient disappearance in back propagating. Compared with Sigmod and Tanh activation functions, the ReLU activation function can also save a lot of computation. The final parameter setting of STMB is summarized in Table VII.

F. Prediction Results

In this work, 90% of the network attack time-series data is adopted as the training set and the remaining 10% as the test set. Figs. 8 and 9 show that the predicted value

TABLE IV
COMPARISON OF DIFFERENT NUMBERS OF HEADS IN STMB

Numbers of heads	RMSLE		MAE		R^2	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
1	0.1443	0.1367	3.0095	3.0065	0.8089	0.8587
2	0.1435	0.0949	3.0042	1.9137	0.8152	0.8854
4	**0.0780**	**0.0886**	**1.6691**	**1.7457**	**0.9460**	**0.9154**
6	0.4399	0.1027	9.8417	2.0537	-0.4349	0.8847

TABLE V
COMPARISON OF DIFFERENT HIDDEN SIZES IN STMB

Hidden sizes	RMSLE		MAE		R^2	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
32	0.0792	0.0923	1.6889	1.8964	0.9418	0.8987
64	0.0811	0.0954	1.7230	1.8982	0.9404	0.8980
72	**0.0780**	**0.0886**	**1.6691**	**1.7457**	**0.9460**	**0.9154**
128	0.0811	0.0943	1.7193	1.8205	0.9407	0.9012

TABLE VI
COMPARISON OF DIFFERENT ACTIVATION FUNCTIONS IN STMB

Activation functions	RMSLE		MAE		R^2	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
Sigmod	0.1154	0.0914	2.1849	1.7807	0.9072	0.8975
Tanh	0.1036	0.0927	2.0618	1.7566	0.9194	0.8960
Softplus	0.1231	0.0898	2.0494	1.7541	0.9089	0.9006
LeakyReLU	0.0830	0.1039	1.9284	1.8667	0.9346	0.8903
ReLU	**0.0780**	**0.0886**	**1.6691**	**1.7457**	**0.9460**	**0.9154**

TABLE VII
PARAMETER SETTING OF STMB

Parameters	Values
Optimizer	Adam
Number of neurons in Bi-LSTM	72
Size of convolutional kernel	9
Number of heads	4
Number of iterations	3000
Window size of the SG filter	5
Polynomial order of the SG filter	3
Coefficients of the SG filter	[-0.08, 0.34, 0.49, 0.34, -0.09]

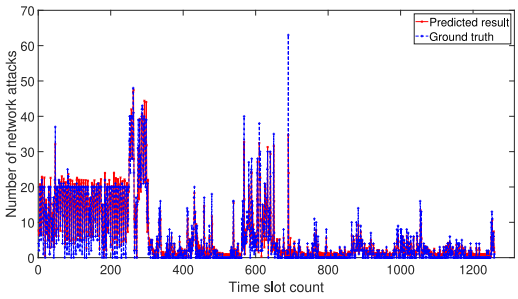


Fig. 9. Prediction results with STMB with data set 2.

has the same increasing and decreasing trends as the ground truth.

To further evaluate the effectiveness and robustness of STMB, RMSLE, MAE, and R^2 are adopted to compare STMB with 16 benchmark models in Table VIII. Each model in the top half of Table VIII does not add the SG filter, and each model in the bottom half applies it. It is shown that STMB achieves the best results. When the SG filter is not adopted,

TABLE VIII
PERFORMANCE COMPARISON OF DIFFERENT METHODS

Methods	RMSLE		MAE		R^2	
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
ARIMA	0.2777	0.3545	6.3833	6.4615	0.4841	0.3710
SVR	0.1516	0.2232	3.1908	3.6790	0.7841	0.6886
BPNN	0.1420	0.2547	2.9904	4.1027	0.8108	0.5054
TCN	0.1444	0.1943	3.0442	2.7089	0.8017	0.7032
LSTM	0.1969	0.1956	2.9494	2.6892	0.8053	0.6931
Bi-LSTM	0.1090	0.1947	2.5309	2.6305	0.8859	0.6954
T-LSTM	0.1459	0.1926	3.0263	2.6086	0.8029	0.7096
T-BiLSTM	0.1043	0.1916	2.4296	2.6057	0.8992	0.7120
TMB	0.1019	0.1905	2.4207	2.6026	0.9007	0.7143
S-ARIMA	0.1801	0.3564	3.7152	3.1203	0.8418	0.5969
S-SVR	0.0934	0.1956	2.0185	2.2176	0.9198	0.8209
S-BPNN	0.0899	0.2086	1.9342	2.7543	0.9218	0.7453
S-TCN	0.0881	0.1024	1.8775	2.0147	0.9277	0.8894
S-LSTM	0.1570	0.0959	2.9494	1.7682	0.8783	0.8953
ST-LSTM	0.0822	0.0936	1.7433	1.7629	0.9399	0.8959
ST-BiLSTM	0.0811	0.0903	1.7327	1.7529	0.9402	0.9023
STMB	**0.0780**	**0.0886**	**1.6691**	**1.7457**	**0.9460**	**0.9154**

RMSLEs of TMB, T-BiLSTM, and T-LSTM are higher than that is adopted in both two data sets. It is shown that the noise in the time series of network attacks is effectively eliminated by the SG filter. Furthermore, the prediction accuracy of TMB is improved. ARIMA and SVM belong to traditional methods, which cannot capture complex, hidden, and nonlinear features among sequences in the series data of network attack, and their experimental results are worse than those of deep learning methods. Both TCN and LSTM have the ability to capture nonlinear features, and therefore, their experimental results are also similar. Our STMB not only has the ability to capture nonlinear features but also assigns different weights to different features, enabling it to better capture correlations in the data. Thus, STMB's prediction accuracy is the highest.

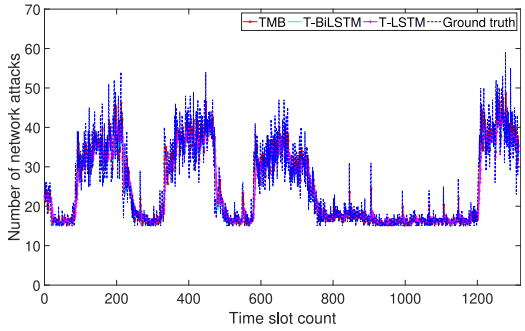


Fig. 10. Prediction results without the SG filter with data set 1.

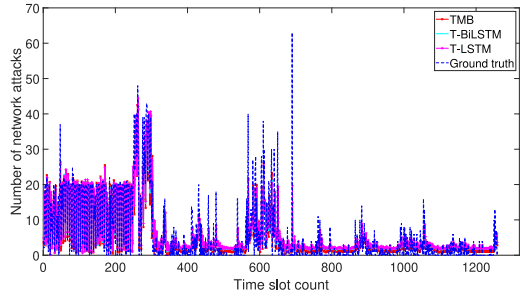


Fig. 11. Prediction results without the SG filter with data set 2.

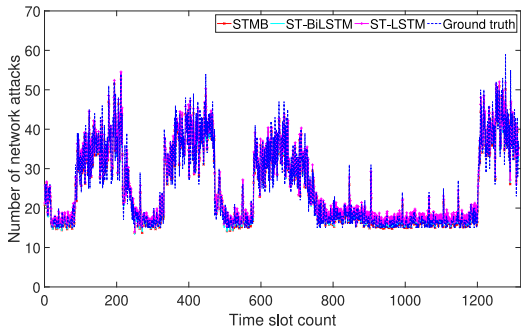


Fig. 12. Prediction results with the SG filter with data set 1.

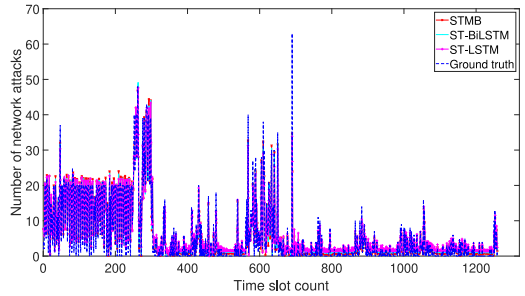


Fig. 13. Prediction results with the SG filter with data set 2.

The predicted and ground-truth values of T-LSTM, T-BiLSTM, and TMB are shown in Figs. 10 and 11. It is shown that the predicted result with TMB and the ground-truth one has the highest fit. During most of the time slots, the predicted values of TMB have smaller errors than other methods and they are closer to the ground-truth values. Figs. 12 and 13 show the predicted values of ST-LSTM, ST-BiLSTM, and STMB. Figs. 14 and 15 show the enlarged prediction results. The results show that three models have

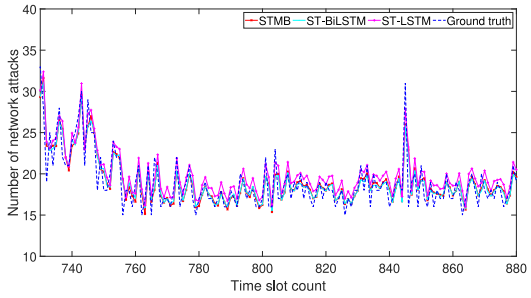


Fig. 14. Enlarged parts of Fig. 12.

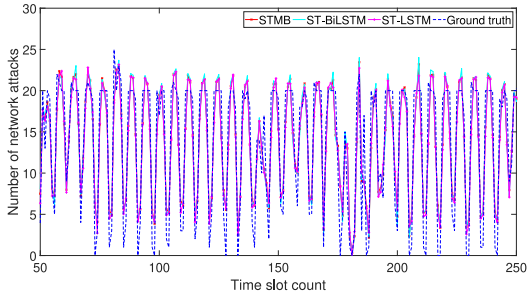


Fig. 15. Enlarged parts of Fig. 13.

higher precision after the smooth processing. During most of the time slots, the predicted value of STMB is closer to the ground-truth one. Thus, STMB achieves the highest prediction accuracy among all models.

V. Conclusion

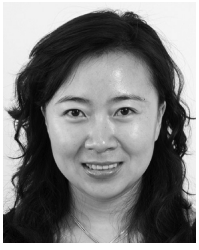
Accurate prediction of network attacks is important for computer network security, network resource management, financial security, etc. However, the number of network attacks exhibits volatile and nonlinear characteristics, making it challenging to predict. This work puts forward a hybrid prediction model called STMB for the first time, which combines the SG filter, TCN, multihead self-attention, and a Bi-GRU network. Specifically, the SG filter is first adopted to remove the noise in the network attack time series. TCN is employed to extract local sequence features. The multihead self-attention is then utilized to capture intrinsic connections among features. Finally, Bi-GRU is adopted to extract bidirectional and long-term correlations in the sequence. Experimental results based on two real-life data sets show that STMB achieves better prediction performance than several state-of-the-art prediction models.

Our future work plans to extend our current work along the following two directions. First, we intend to improve our filter to better handle the curve of a network attack sequence and to eliminate its noise data. Second, we intend to adopt more efficient neural networks [40] to reduce the execution time and consumed memories during the model training.

References

[1] 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.

[2] J. Bi, H. Yuan, and M. Zhou, "Temporal prediction of multiapplication consolidated workloads in distributed clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1763–1773, Oct. 2019.
[3] M. Choraś and R. Kozik, "Machine learning techniques applied to detect cyber attacks on Web applications," *Logic J. IGPL*, vol. 23, no. 1, pp. 45–56, Feb. 2015.
[4] G. Franzè, F. Tedesco, and D. Famularo, "Resilience against replay attacks: A distributed model predictive control scheme for networked multi-agent systems," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 3, pp. 628–640, Mar. 2021.
[5] E. C. Hall, G. Raskutti, and R. M. Willett, "Learning high-dimensional generalized linear autoregressive models," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2401–2422, Apr. 2019.
[6] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003.
[7] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct.–Dec. 2015.
[8] Y. Wang et al., "Short-term load forecasting for industrial customers based on TCN-LightGBM," *IEEE Trans. Power Syst.*, vol. 36, no. 3, pp. 1984–1997, May 2021.
[9] Z. Geng, Z. Chen, Q. Meng, and Y. Han, "Novel transformer based on gated convolutional neural network for dynamic soft sensor modeling of industrial processes," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1521–1529, Mar. 2022.
[10] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.
[11] J. Bi, K. Xu, H. Yuan, and M. Zhou, "A hybrid deep learning method for network attack prediction," in *Proc. Int. Conf. Systems, Man, Cybern.*, Prague, Czech Republic, 2022, pp. 544–549.
[12] H. Moudoud, L. Khoukhi, and S. Cherkaoui, "Prediction and detection of FDIA and DDoS attacks in 5G enabled IoT," *IEEE Netw.*, vol. 35, no. 2, pp. 194–201, Mar./Apr. 2021.
[13] K. Huang, C. Zhou, Y. C. Tian, S. Yang, and Y. Qin, "Assessing the physical impact of cyberattacks on industrial cyber-physical systems," *IEEE Trans. Ind. Electron.*, vol. 65, no. 10, pp. 8153–8162, Oct. 2018.
[14] A. Bar, B. Shapira, L. Rokach, and M. Unger, "Identifying attack propagation patterns in honeypots using Markov chains modeling and complex networks analysis," in *Proc. IEEE Int. Conf. Software Sci., Technol. Eng.*, 2016, pp. 28–36.
[15] G. Werner, S. Yang, and K. McConky, "Time series forecasting of cyber attack intensity," in *Proc. 12th Annu. Conf. Cyber Inf. Secur. Res.*, New York, NY, USA, 2017, pp. 1–3.
[16] A. Okutan, G. Werner, K. McConky, and S. J. Yang, "POSTER: Cyber attack prediction of threats from unconventional resources (CAPTURE)," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2017, pp. 2563–2565.
[17] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and ARIMA model in cloud data center," in *Proc. IEEE 15th Int. Conf. Netw., Sensing Control*, Zhuhai, China, 2018, pp. 1–6.
[18] K.-D. Lu, G.-Q. Zeng, X. Luo, J. Weng, W. Luo, and Y. Wu, "Evolutionary deep belief network for cyber-attack detection in industrial automation and control system," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7618–7627, Nov. 2021.
[19] A. Al-Abassi, H. Karimipour, A. Dehghantanha, and R. M. Parizi, "An ensemble deep learning-based cyber-attack detection in industrial control system," *IEEE Access*, vol. 8, pp. 83965–83973, 2020.
[20] P. Ganesh et al., "Learning-based simultaneous detection and characterization of time delay attack in cyber-physical systems," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3581–3593, Jul. 2021.
[21] J. Gao, H. Sultan, J. Hu, and W.-W. Tung, "Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: A comparison," *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 237–240, Mar. 2010.
[22] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964.
[23] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.
[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, Jan. 2014.
[25] A. Vaswani et al., "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–11.
[26] Y. Li, S. Wang, Y. Wei, and Q. Zhu, "A new hybrid VMD-ICSS-BiGRU approach for gold futures price forecasting and algorithmic trading," *IEEE Trans. Comput. Social Syst.*, vol. 8, no. 6, pp. 1357–1368, Dec. 2021.
[27] Y.-J. Gong et al., "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.
[28] S. Lee and S. B. Kim, "Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1157–1166, Jun. 2020.
[29] C. Peng, X. Wu, W. Yuan, X. Zhang, Y. Zhang, and Y. Li, "MGRFE: Multilayer recursive feature elimination based on an embedded genetic algorithm for cancer classification," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 18, no. 2, pp. 621–632, Mar. 2021.
[30] S. Jachner, K. G. V. D. Boogaart, and T. Petzoldt, "Statistical methods for the qualitative assessment of dynamic models with time delay (R Package qualV)," *J. Stat. Softw.*, vol. 22, no. 8, pp. 1–30, Sept. 2007.
[31] 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, Jul.–Sep. 2022.
[32] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, pp. 1–15.
[33] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," in *Proc. IEEE 4th Int. Telecommun. Netw. Workshop QoS Multiservice IP Netw.*, Venice, Italy, 2009, pp. 191–201.
[34] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "RVLBPNN: A workload forecasting model for smart cloud computing," *Sci. Program.*, vol. 2016, pp. 1–9, Nov. 2016.
[35] C. Ma, G. Dai, and J. Zhou, "Short-term traffic flow prediction for urban road sections based on time series analysis and LSTM_BILSTM method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5615–5624, Jun. 2022.
[36] 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.
[37] H. Yuan, Q. Hu, and J. Bi, "Cost-minimized and multi-plant scheduling in distributed industrial systems," in *Proc. IEEE Int. Conf. Systems, Man, Cybern.*, Prague, Czech Republic, 2022, pp. 2851–2856.
[38] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.
[39] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, vol. 424, pp. 35–48, Feb. 2021.
[40] X. Hou, K. Wang, C. Zhong, and Z. Wei, "ST-Trader: A spatial-temporal deep neural network for modeling stock market movement," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 5, pp. 1015–1024, May 2021.



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 the 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 currently 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.

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

Kangyuan Xu (Member, IEEE) received the B.S. degree in software engineering from Hunan Normal University, Changsha, China, in 2021. He is currently pursuing the master's degree with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China.

His research interests include cloud computing, data center, energy management, big data, time-series prediction, machine learning, and deep learning.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology, 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 Best Poster Prize in the 3rd IFAC Workshop on Cyber-Physical and Human Systems 2020 and the Best Paper Award in the 17th ICNSC.

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

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair of Engineering and a Professor with 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

In 1990, he joined New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has over 900 publications, including 12 books, 600+ journal papers (450+ in IEEE transactions), 28 patents, and 29 book chapters. His interests are in Petri nets, automation, Internet of Things, and big data.

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