Wang *et al.*
EURASIP Journal on Information Security (2023) 2023:5
https://doi.org/10.1186/s13635-023-00139-y

EURASIP Journal on Information Security

RESEARCH

Open Access

Network intrusion detection based on multi-domain data and ensemble-bidirectional LSTM

Xiaoning Wang[1], Jia Liu[2] and Chunjiong Zhang[3*]

Abstract

Different types of network traffic can be treated as data originating from different domains with the same objectives of problem-solving. Previous work utilizing multi-domain machine learning has primarily assumed that data in different domains have the same distribution, which fails to effectively address the domain offset problem and may not achieve excellent performance in every domain. To address these limitations, this study proposes an attention-based bidirectional long short-term memory (Bi-LSTM) model for detecting coordinated network attacks, such as malware detection, VPN encapsulation recognition, and Trojan horse classification. To begin, HTTP traffic is modeled as a series of natural language sequences, where each request follows strict structural standards and language logic. The Bi-LSTM model is designed within the framework of multi-domain machine learning technologies to recognize anomalies of network attacks from different domains. Experiments on real HTTP traffic data sets demonstrate that the proposed model has good performance in detecting abnormal network traffic and exhibits strong generalization ability, enabling it to effectively detect different network attacks simultaneously.

Keywords Anomaly detection, Attention mechanism, Bidirectional LSTM, Multi-domain learning

1 Introduction

The Ethernet protocol has been widely used in recent decades. One of the most significant properties of Ethernet protocols is that they are partially open source and designed with security policy flaws [1], which means that network equipment is vulnerable to serious security threats such as viruses, Denial of Service (DoS) attacks, and port cracking access. As software systems and applications become increasingly complex, more and more vulnerabilities have emerged, causing great harm to network security [2]. Network intrusion detection is a means of network protection, which aims to analyze the network traffic, log data, and related information and discover abnormal traffic data from a large set of data streams. Once the network traffic is recognized as abnormal, it can be considered as an attack on the network so that related measures must be taken in advance to avoid catastrophic loss [3].

Basically, existing network anomaly detection methods can be divided into two categories: one is based on feature distribution [4, 5], and the other is based on content detection [6, 7]. However, these methods often lack the ability to distinguish between different types of attacks, leading to inaccurate detection of some anomalies. Recent studies have proposed using machine learning techniques to discover abnormal behavior from HTTP server logs and cluster HTTP session processes using a non-parametric hidden Markov model with explicit state duration [8]. Additionally, some researchers have used unsupervised deep belief networks (DBNs) to extract low-level features and trained single-class support vector machines (SVMs) to perform anomaly detection and

*Correspondence:
Chunjiong Zhang
gip_1986@126.com
[1] Chongqing Vocational Institute of Tourism, Chongqing 409099, China
[2] Chongqing College of Electronic Engineering, Chongqing, China
[3] College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

diagnosis from network traffic [9]. These methods usually only involve the detection of a single type of traffic and cannot automatically extract and identify the key features of multiple types of traffic [10].

These models can automatically discover key features in HTTP traffic before learning complex attack patterns, making them particularly useful for detecting malicious network traffic. However, many network attackers often conduct coordinated attacks between different types of traffic due to the complexity of the network environment. To address this challenge, researchers have proposed using multi-domain machine learning techniques to improve the detection efficiency by sharing solving processes across different domains. This approach has shown acceptable anomaly detection capabilities for diverse types of attacks, making it a promising area of research for improving network security [11].

However, most deep learning models simply assume that data captured from different domains have the same distribution, which can neither effectively address the domain offset problem, nor achieve excellent performance in every domain [12]. One reason why this happens is that it is difficult for most models to automatically detect hidden abnormal traffic from a large amount of network traffic. The other reason is that the lack of an integrated end-to-end system makes it impossible to discover the characteristics of traffic in different domains at the same time. To this end, we firstly propose a bidirectional long-short-term memory (Bi-LSTM) model based on the embedded attention mechanism to simultaneously detect different types of coordinated network attacks (i.e., malware detection, VPN encapsulation recognition, and Trojan Horse classification). Then, we designed a multi-domain neural network to detect abnormal traffic processes from different domains. The elements in the traffic are modeled as vocabulary, and the semantic relationship between vocabularies can be learned by the proposed Bi-LSTM. Experiments demonstrate that the model proposed in this study has high precision and generalization ability on the real HTTP traffic data set.

2 Related work

For the purpose of network intrusion detection, machine learning approaches such as cluster analysis, Bayesian networks, particle algorithms, and other shallow structure algorithms are commonly used [10, 11]. Most of the algorithms can learn enriched features of the data, but they are difficult to be used in practical scenarios due to different means of intrusion and heterogeneous distributions of the data. Network intrusion detection based on deep learning is independent of feature engineering and can automatically learn complex features from raw traffic data [13]. Therefore, many scholars have applied neural networks to network intrusion detection tasks. Due to the improvement of the traffic classification accuracy, network anomaly detection based on neural networks has become a hot research topic in academia. Representative examples with respect to these studies are as follows. The authors in the literature [14] first mapped network traffic features to strings and translated the network traffic classification problem to text classification in natural language processing. They used recurrent neural networks (RNN) to learn temporal features of network traffic and further used it for malicious network traffic detection. The authors in the literature [15] proposed a malware traffic classification algorithm based on convolutional neural networks (CNN) for intrusion detection by mapping traffic features to generate grayscale pixel images. They processed network intrusion detection with network traffic image classification. The authors in the literature [16] made a relevant summary of deep learning models, focusing on traffic data simplification, dimensionality reduction, and classification techniques. They proposed fully convolutional networks, which proved the effectiveness in analyzing the network traffic. In addition, the authors in the literature [17] proposed an anomalous traffic detection method based on long-short-term memory (LSTM) with improved residual neural network optimization, which addressed the disadvantages such as overfitting and gradient disappearance in deep neural networks. As a result, the proposed model improved the accuracy of network anomalous traffic detection.

As a matter of fact, the CNN and RNN sequence models that combine attention mechanism with deep learning have played a critical role in the field of anomaly detection [9–11]. CNN has a generalization ability to a variety of classification tasks and can obtain the spatial characteristics of data during intrusion detection. However, the poor convergence of model training speed usually results in low accuracy and high false alarm rate [18]. RNN extracts time series and semantic information using temporal information as input; however, the training process costs too much compared to the shallow structure algorithms [19]. Although the auto-encoder can be used to capture the nonlinear information in the data by traditional dimension-reduction methods, the model evaluation index is too simple to improve the error rate of detection [20].

Network intrusion detection based on multi-domain network traffic considers the multi-domains of data. Because few classes of samples are likely to make the model overfitting during the training process and the model prediction is more biased to the majority of samples, the previous multi-domain method is less accurate

for malicious attack identification than existing CNN and RNN-based approaches. Therefore, many scholars are motivated to study the multi-domain distribution of network traffic data. For example, the authors in the literature [21] proposed a method to improve the accuracy of minority class classification. The method combines the minority oversampling technique (SMOTE) and complementary neural network (CMTNN) to solve the multi-domain problem caused by network traffic distribution. Experiments on the UCI data set report that the proposed method can improve the intrusion detection accuracy and identification of multi-domains. The authors in the literature [22] proposed an improved local adaptive composite minority sampling algorithm (LA-SMOTE) to deal with the network traffic multi-domain problem and then detect network traffic anomalies based on a gated recurrent unit neural network. The authors in the literature [23] combined upsampling and downsampling methods to deal with the multi-domain data set CIDDS-001 and evaluated the data set with classifiers such as deep neural networks, random forests, and variational autoencoders. Recently, deep autoencoders are used to build data generation models to form balanced data sets [24]. However, the authors in the literature [25] proposed a Siamese intrusion detection system based on twin neural networks to detect R2L and U2R attacks without using traditional class balancing techniques. The use of a generative adversarial network (GAN) model is proposed to generate samples with different attack labels (e.g., blacklisting, anomalous spam, SSH brute force cracking) for balancing the data set, which improves the classification accuracy [26]. To summarize, although the related works contribute to network anomaly detection, the heterogeneous data distribution caused by the multi-domain problem makes the proposed algorithms little effective, especially in the detection of attacks with few class samples.

3 Proposed model

The structure of the proposed model is shown in Fig. 1. The left side of Fig. 1 is the Bi-LSTM module based on the attention mechanism, which consists of four parts, i.e., the input layer, the embedding layer, the Bi-LSTM layer, and the attention layer. The Bi-LSTM module provides specific knowledge for the individual layer. We train the content sequence and adopt an exit strategy to avoid overfitting before it comes to the output layer. The right side of Fig. 1 shows three Bi-LSTM modules that connect the output of the attention layer to multi-domain learning and jointly train a Bi-LSTM hybrid model with integrated attention in the multi-domain learning framework. The three-domain tasks share the parameters of the lower layers and provide prior knowledge for the public layer.

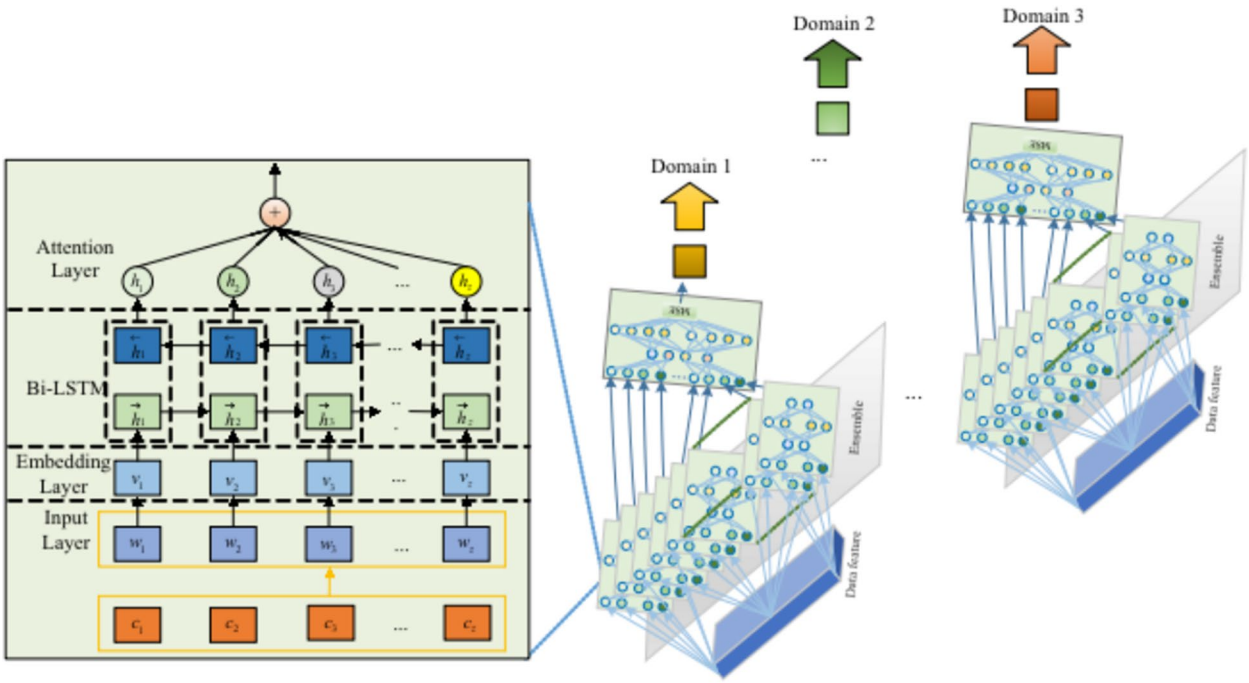

Fig. 1 Structure of the proposed model

3.1 Ensemble Bi-LSTM

3.1.1 Input layer

This study selects common vocabulary in the training set to build a vocabulary based on term frequency-inverse document frequency (TF-IDF) [27] and generates a unique index for each vocabulary. The final input vector consists of a unique word identifier of a fixed length Z, where Z is a hyperparameter. The redundant part of the data is removed, and the insufficient part is filled with zeros. For example, given a series of content: {'/', 'admin', '/', 'caches', '/', 'error_ches', '', 'php'}, its input vector can be represented by [23, 3, 23, 56, 23, 66, 0, 0, ..., 0] because the index of "/" in the vocabulary is 23. Since the length of the input vector is less than the fixed value, the remaining part is filled with zeros.

3.1.2 Embedding layer

We convert the original input sequence into an input vector, denoted as $s_i = \{w_1, w_2, w_3, \ldots, w_z\}$, and obtain an embedding vector $\mathbf{v} = \{v_1, v_2, v_3, \ldots, v_z\}$. Then, the vector of each word is represented as Eq. (1).

$$v_k = \mathrm{ReLU}(W_e w_k + b_e) \in R^m \tag{1}$$

where $w_k \in R^1, (k = 1, 2, \ldots, z)$ is the input vector, $W_e \in R^{m \times 1}$ is the weight matrix, $b_e \in R^m$ is the bias vector, and m is the size of the embedding dimension and ReLU is the activation function [18].

3.1.3 Bi-LSTM layer

Bi-LSTM is composed of forward LSTM and backward LSTM, which can use previous and future information to improve prediction performance. Given the embedding vector $\underset{\rightarrow}{\mathbf{v}} = \{v_1, v_2, v_3, \ldots, v_z\}$ of request R_i, the forward LSTM $\overrightarrow{f}$ reads the input sequence from v_1 to v_z, and calculate the forward hidden state sequence $\left(\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_z\right)\left(\overrightarrow{h}_i \in R^p\right)$, where p is the number of dimensions of the hidden state. The backward LSTM $\overleftarrow{f}$ reads the input sequence in reverse order and generates a backward hidden state sequence $\left(\underset{1}{\overleftarrow{h}}, \underset{2}{\overleftarrow{h}}, \ldots, \underset{z}{\overleftarrow{h}}\right)\left(\underset{i}{\overleftarrow{h}} \in R^p\right)$. Concatenating the forward hidden state $\overrightarrow{h}_i$ and the backward hidden state $\underset{i}{\overleftarrow{h}}$, the final latent vector can be represented as $h_i = \left[\overrightarrow{h}_i; \underset{i}{\overleftarrow{h}}\right]^{\mathrm{T}} \left(h_i \in R^{2p}\right)$.

3.1.4 Attention layer

The attention mechanism usually only considers the hidden state information of the individual and cannot capture the relationship between the current hidden state and previous hidden states. This study uses location-based attention to capture the relationship between h_z and $h_i (1 \le i < z)$. The attention weight is calculated as Eq. (2):

$$v_k = \mathrm{ReLU}(W_e w_k + b_e) \in R^m \tag{2}$$

where $W_\alpha \in R^{2p}$ and $b_\alpha \in R$ are model parameters and bias, respectively.

The weight vector α_z is expressed as Eq. (3):

$$\alpha_z = \mathrm{softmax}\left(\left[\alpha_{z1}, \alpha_{z2}, \ldots, \alpha_{z(z-1)}\right]\right) \tag{3}$$

where $\alpha_{zi} = v_\alpha^T \tanh\left(W_\alpha [h_z; h_i]\right)$. The context vector c_z [19] is represented as Eq. (4):

$$c_z = \sum_i^{z-1} \alpha_{zi} h_i \tag{4}$$

The hidden state $\tilde{h}$ can be calculated as Eq. (5):

$$\tilde{h} = \tanh\left(W_c [c_z; h_z]\right) \tag{5}$$

where $W_c \in R^{r \times 4p}$ is the weight matrix in the attention layer, and r is the number of dimensions of the state in the attention layer.

3.2 Multi-domain learning

3.2.1 Proof of feasibility

To some extent, the feasibility of multi-domain learning is determined by its ability to prevent overfitting. Overfitting means that the model can accurately learn the information on the training data set but cannot handle the random test set. Generally speaking, a more complex model may suffer from more severe overfitting.

For a single learning task, its cost function $J(\cdot)$ can be represented by Eq. (6):

$$J\left(\theta_s, \theta_i\right) = \frac{1}{2n} \sum_{i=1}^{n} \left(F_{\theta_s}(\tilde{h}) + F_{\theta_i}(\tilde{h}) - y\right) = F\left(\tilde{h}; \theta_s, \theta_i\right) \tag{6}$$

where $F(\cdot)$ represents the objective function of multi-domain machine learning, y is the label, $\tilde{h}$ represents the use of hidden state of different channels as the input of the multi-domain, θ_s is the shared parameter set with the best value Θ_s, and θ_i is the parameter set with the best value Θ_i for the single task.

Minimizing the weighted sum of $J(\cdot)$, we get Θ_i and Θ_s, formulated as Eq. (7):

$$\left(\Theta_s, \Theta_i\right) = \mathrm{argmin}_{\theta_s, \theta_i} \sum_{i=1}^{l} \alpha_i F\left(\tilde{h}; \theta_s, \theta_i\right) \tag{7}$$

where α_i is the discount factor of θ_i and l is the number of domains.

Since multi-domain learning uses different θ_i, the loss function R_i of tasks in different domains can be regarded as the regularization function of Eq. (8):

$$\begin{aligned}(\Theta_s, \Theta_i) &= \text{argmin}_{\theta_s,\theta_i}\left[\alpha_i F(\tilde{h};\theta_s,\theta_i)\right] + \sum_{j\neq i}^{n} \alpha_j F(\tilde{h};\theta_s,\theta_j) \\ &= \text{argmin}_{\theta_s,\theta_i}\left[\alpha_i F(\tilde{h};\theta_s,\theta_i)\right] + \lambda R_i(\tilde{h};\theta_s)\end{aligned} \tag{8}$$

Regularization is the most common and effective method to deal with overfitting. Adding a regularization function to the loss function can simplify its high-dimensional structure. Therefore, the optimal shared parameters are restricted to a smaller solution, making the learner Θ_s equally effective for other tasks.

3.2.2 The loss functions for different attacks

Since malware detection and VPN encapsulation identification are processed in a common fully connected layer, and the data of Trojan Horses needs to be fed to an additional layer, this study designs different loss functions for different network attacks.

Since the amount of data detected by malware is small, it can be treated as a binary classification problem. The loss function based on cross-entropy is suitable for this task. For VPN encapsulation identification, it can be regarded as a regression problem in different scenarios. The loss function of the binary classification problem is shown in Eq. (9):

$$f_1 = \text{mlog}\left(\frac{1}{p_m}\right) + (1-m)\log\left(1-\frac{1}{p_m}\right) \tag{9}$$

where $m = 1$ represents VPN encapsulation, $m = 0$ represents normal data, and p_m represents the predicted probability that the data segment will be encapsulated by the VPN.

The loss function of the regression problem is shown in Eq. (10):

$$f_2 = \lambda\frac{1}{2}(p_y - s)^2 + (1-\lambda)\left[1 - \exp\left(\frac{-(p_y-s)^2}{2\sigma^2}\right)\right] \tag{10}$$

where p_y is the suspicious degree of VPN encapsulation attacks, s is the actual degree of suspiciousness, σ^2 is the variance of the suspicious degree between p_y and s, $\frac{1}{2}(y-s)^2$ is the loss of Euclidean function, and $1-\exp\left(\frac{-(y-s)^2}{2\sigma^2}\right)$ is the loss of Gaussian function. When the initial prediction value is far from the actual value, the loss effect of the Euclidean function is better than the loss of the Gaussian function. By contrast, the loss of the Gaussian function is more useful when σ^2 degrades dramatically [28]. We obtain a better learning ability and anomaly detection performance of the VPN encapsulation model by changing values of λ.

For the Trojan Horse program, the multi-type cross-entropy is used as the loss function, as shown in Eq. (11):

$$f_3 = \sum_{i=0}^{z} y_i \log\left(\frac{1}{p_i}\right) \tag{11}$$

Let F be the total loss of multi-domain machine learning. The loss functions of anomaly detection tasks in different domains can be unified by Eq. (12):

$$F = \beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3 \tag{12}$$

where β_i is the discount factor corresponding to the task i. Since the loss gradient of regression tasks in the learning process is often smaller than that of classification tasks, we set β_2 to a large value constrained to $\beta_1 + \beta_2 + \beta_3 = 1$.

3.3 Convergence analysis

Regarding the convergence of the model, it can be analyzed by the parameter update process of the incoming gradient. Since the batch data normalization layer normalizes the input, it needs the trainable parameters in the layer to maintain the expressiveness of the network. By updating the trainable parameters in the batch data normalization layer using local gradients, the degradation of model expressiveness caused by sparse gradients can be prevented. The model is updated with sparse gradients in a way that makes full use of the data to prevent the loss of model accuracy due to sparsity [29].

During model training, the gradient sparsity is updated in such a way that each domain data is sparse locally for the current generated gradient $\mathbf{g}_t$ uploaded and averaged by the parameter server. The gradient is adaptively computed by the optimizer and multiplied by the learning rate α for updating the parameters. This process can be expressed by Eq. (13).

$$\boldsymbol{\theta}_{t+1}^{k} = \boldsymbol{\theta}_{t}^{k} - \alpha \cdot \frac{\phi_t\left(\sqrt{\overline{\text{sparse}(\mathbf{g}_1)}}, \overline{\text{sparse}(\mathbf{g}_2)}, \ldots, \overline{\text{sparse}(\mathbf{g}_t)}\right)}{\sqrt{\varphi_t\left(\overline{\text{sparse}(\mathbf{g}_1)}, \overline{\text{sparse}(\mathbf{g}_2)}, \ldots, \overline{\text{sparse}(\mathbf{g}_t)}\right)}} \tag{13}$$

where α represents the average of the sparse gradient of all domain data uploads, i.e.:

$$\overline{\text{sparse}(\mathbf{g}_t)} = \frac{1}{N} \cdot \sum_{k=1}^{N} \text{sparse}(\mathbf{g}_{k,t}) \tag{14}$$

After using the gradient correction, the domain data is computed by the optimizer for the complete gradient.

The sparse result for that gradient is uploaded and collected by the neural network layer with shared parameters. So, the parameter updates after using the gradient correction can be expressed by Eq. (15).

$$\theta_{t+1}^{k} = \theta_{t}^{k} - \frac{\alpha}{N} \cdot \sum_{k=1}^{N} \text{sparse}\left(\frac{m_{k,t}}{\sqrt{V_{k,t}}}\right) \tag{15}$$

where $\mathbf{m_t} = \phi_t(\overline{\mathbf{g_1}}, \ldots, \overline{\mathbf{g_t}})$ and $\mathbf{V_t} = \varphi_t(\overline{\mathbf{g_1}}, \ldots, \overline{\mathbf{g_t}})$, N is the number of samples.

The update is performed with sparse gradients for all trainable parameters in the model. For the output channels of the neural network layers, the group normalization is formed as a group to further improve the batch normalization. The batch normalization, layer normalization, instance normalization, and group normalization differentiate the range of the set S_i. We use T to denote the number of feature maps, C to denote the number of channels, and $H \times W$ to denote the size of the feature map. Then, we have Eq. (16).

$$S_i = \{k | k_C = i_C\} \tag{16}$$

The layer normalization is the set of all feature maps in which only the points on the same feature map constitute can participate in the calculation of the mean and standard deviation of the current point. The set is represented by Eq. (17):

$$S_i = \{k | k_T = i_T\} \tag{17}$$

The instance normalization is the set of all features that only the set of points on the same feature map can participate in the calculation of the current point mean and standard deviation. The set is represented by Eq. (18):

$$S_i = \{k | k_T = i_T, k_C = i_C\} \tag{18}$$

The group normalization is caused by grouping in the channel dimension. Suppose that the number of channels C is divided into G groups, then the set is represented by Eq. (19):

$$S_i = \left\{k | k_T = i_T, \left\lfloor \frac{k_C}{C_G} \right\rfloor = \left\lfloor \frac{i_C}{C_G} \right\rfloor \right\} \tag{19}$$

where $\left\lfloor \frac{k_C}{C_G} \right\rfloor = \left\lfloor \frac{i_C}{C_G} \right\rfloor$ indicates the points in the same group.

While training the model, the parameters to be synchronized in the batch data normalization layer consist of the mean μ_B and the variance σ_B^2. Since they refer to the results of the current batch, it is necessary to communicate with each other to transmit the results calculated from the local data. The optimal solution is to transmit the square of the mean to each other because it can compute the mean and variance of the overall batch data accurately with minimal communication overhead.

In the gradient-efficient update method, all trainable parameters in the model are updated with sparse gradients, which results in a forced delay in the update of most gradients. In order to alleviate the delay, the full set of gradient update parameters is considered to be utilized.

$$\boldsymbol{\theta}_{t+1}^{k} = \boldsymbol{\theta}_{t}^{k} - \frac{\alpha}{N} \cdot \left(\frac{\mathbf{m}_{j,t}}{\sqrt{V_{j,t}}} + \sum_{k=1}^{N(k \neq j)} \text{sparse}\left(\frac{\mathbf{m}_{k,t}}{\sqrt{V_{k,t}}} \right) \right) \tag{20}$$

where $\boldsymbol{\theta}_{t+1}^{k}$ is a parameter on the kth domain when $t+1$ training iterations are performed. At this moment, the parameter update uses the complete result computed by the optimizer and the flatness transmitted by the other domains. Compared with the original update method, the sparse gradient is not used in the part of the gradient generated by the node itself. Herein, the complete gradient is used instead [30].

It should be noted that, when the data distribution is changed, the true distribution of the training data is not well presented. To solve this problem, the batch data normalization layer has two trainable parameters, which are used to change the magnitude of the local variance and the mean position of the data. The changed distribution can be more consistent with the real distribution to facilitate the nonlinear expression of the model. The two trainable parameters are closely related to the current input, so we consider adjusting the parameter update method with Eq. (21).

$$\boldsymbol{\theta}_{t+1}^{k} = \begin{cases} \boldsymbol{\theta}_{t}^{k} - \frac{\alpha}{N} \cdot \sum_{k=1}^{N} \text{sparse}\left(\frac{\mathbf{m}_{k,t}}{\sqrt{\mathbf{V}_{k,t}}} \right), \text{ for } \boldsymbol{\theta} \notin BN \\ \boldsymbol{\theta}_{t}^{k} - \frac{\alpha}{N} \cdot \left(\frac{\mathbf{m}_{j,t}}{\sqrt{\mathbf{V}_{j,t}} +} + \sum_{k=1}^{N(k \neq j)} \text{sparse}\left(\frac{\mathbf{m}_{k,t}}{\sqrt{V_{k,t}}} \right) \right), \text{ for } \boldsymbol{\theta} \in BN \end{cases} \tag{21}$$

For those parameters that are not in the batch data normalization layer, each node updates the parameters using the average sparse gradient sent down from the shared neural network layer. For those parameters that belong to the batch data normalization layer, different domains use the local complete gradient instead of the sparse gradient generated by themselves, which is computed together with the sparse gradient passed from other nodes. The gradient of the in-domain part that is used to update the parameters in the batch data normalization layer is always complete without delay. The sparse and accumulative process only serves other domains besides itself.

In the sparse case, we find that the multi-domain approach proposed in this paper performs better in the model enriched with batch data normalization layers compared to the traditional approach of using sparse

Table 1 Data of HTTP abnormal traffic

Categories	Number of entries
File inclusion	46,438
SQL injection	163,776
Webshell	288,050
XSS	127,750
Sensitive data exposure	166,56
Struts2 vulnerabilities	42,477
Total	685,147

Table 3 Data selected from ISCX

Way of communication	Content
VPN chat	AIM, Facebook, ICQ, and Skype
Chat	
VPN e-mail	Gmail
E-mail	
VPN traffic	Inetflix, Spotify, Sopcast, and Skype
Traffic	
VPN-P2P	Bittorrent and uTorrent
P2P	

gradients from all nodes. The parameters of the batch data normalization layer of the domain itself can be applied directly and locally mitigating the convergence bias caused by delayed updates.

4 Experiment

4.1 Data set

4.1.1 HTTP traffic data set

The dataset used in this experiment, the HTTP traffic data set, is a collection of real traffic generated during vulnerability scanning [31]. The dataset includes 685,147 abnormal traffic events, which are categorized into six types: file inclusion, SQL injection, webshell, cross-site scripting (XSS), sensitive data exposure, and Struts2 vulnerabilities, as shown in 1. Additionally, the dataset also includes 1 million normal and unmarked traffic samples for testing purposes. This diverse range of traffic patterns allows researchers to evaluate the performance of their proposed models on different types of network traffic and identify patterns that may be specific to certain types of attacks or vulnerabilities.

4.1.2 CTU-13 data set

CTU-13 data set [21], developed in 2011 by cyber security researchers at the Czech Technical University in Prague, is a comprehensive dataset that includes various types of malware traffic. The study focuses on evaluating the performance of the proposed model by selecting eight representative malwares from the CTU-13 data set and analyzing their behavior patterns, as shown in Table 2. This approach allows for a more detailed assessment of the model's capabilities in detecting and mitigating malware attacks, which are increasingly prevalent in today's digital landscape.

4.1.3 ISCX data set

ISCX [22] was developed by Draper Gil et al. in 2015 and comprises seven common traffic types as well as their corresponding VPN encapsulations, totaling a staggering fourteen categories. In this study, a selection of eight categories from the aforementioned list was chosen, as evidenced by Table 3.

4.1.4 KDDcup99

This dataset was utilized in the third International Knowledge Discovery and Data Mining Tools Competition, which took place concurrently with the 5th International Conference on Knowledge Discovery and Data Mining (KDD-99). The competition task involved building a network intrusion detector, which is a predictive model that can differentiate between "bad" connections (referred to as intrusions or attacks) and "good" normal connections. The database contains a set of standard data for auditing purposes, including various types of simulated intrusions in a military network environment. Its multi-domain division can be categorized into R2L, U2B, DoS, and Probe.

Table 2 Eight typical malwares in CTU-13

CTU serial number	Name of software	Explanation
43	Neris	Famous malware captured in August 2011
114–3	Emotet	Banking Trojan malware, captured from April to June 2015
116–1	Kazy	A widely spread Trojan Horse, captured in May 2012
119–2	Geodo	Banking Trojan malware, captured from April to June 2015
127–1	Miuref	Troian captured in June 2015 for Windows
142–1	Shifu	Japanese Banking Trojan Horse, captured in September 2015
147–1	Avzhan	DDOS robot, taken in September 2015
158–1	Tinna	Trojan Banking taken in April 2016

4.1.5 CCCS-CIC-AndMal2020

TheAndroid malware dataset, designated as CCCS-CIC-AndMal-2020,comprises 200,000 benign and 200,000 malicious software samples. Thisvast collection of data includes a total of 4 million Androidapplications, spanning across 14 renowned malware categories and 191famous malware programs. The multi-domain nature of this dataset canbe effectively represented using different types of malware.

4.2 Experimental settings

This experiment utilizes the Windows 10 64-bit operating system, equipped with a 18GB RAM and a quad-core Intel Core i7-10900 processor, as well as Nvidia GTX1080 GPU acceleration. The Python framework for implementation is TensorFlow and Keras [5]. The optimization algorithm employed is ADAM [12], with a batch size of 200, input training dataset size of 96,000 samples, and flow rate per type of traffic set at 6,000. Additionally, another 64,000 samples were selected for testing purposes, with each type of abnormal traffic having 4,000 samples.

In this experiment, each entry in the dataset is treated as an array and converted into text for LSTM learning. Initially, we train our model using a labeled HTTP traffic dataset. Our evaluation metrics include precision, recall, and F1 score, which enable us to assess the model's ability to detect anomalies in HTTP traffic. Subsequently, we validate the model's performance on an additional 1 million unlabeled HTTP traffic datasets. Furthermore, this experiment demonstrates the effectiveness of our model for detecting network anomalies across multiple domains using the CTU-13 and ISCX datasets. The evaluation metrics used are precision and recall.

4.3 Comparison algorithms

This experiment uses the following comparison algorithms for HTTP traffic anomaly detection.

(1) Deep feed-forward artificial neural networks (CNN), which is widely used in image processing
(2) RNN, often used for time series prediction and speech recognition
(3) Bi-LSTM, which can predict each feature of the sequence based on the previous state and context
(4) The following comparison algorithms are used in multi-domain anomaly detection
(5) Parallel cross-CNN [22] (PCCN), fusing different domain characteristics from branches of CNN
(6) A hybrid of feed-forward neural networks (FNN) and CNN [23] (HDCM), where FNN is used to process formatted data, thereby reducing the overall complexity of learning
(7) Decision tree which is an integrated learning scheme
(8) Single-task learning (STL) of the basic layer of the model proposed in this study.

5 Results

5.1 Detection of labeled data sets

The results of the labeled HTTP data set indicate that the recall rate and F1 score of the sequence model (LSTM and Bi-LSTM) are superior to those of CNN. In particular, Bi-LSTM has a better performance than LSTM. The model proposed in this study outperforms both the sequence model and CNN in terms of precision, recall, and F1 score, as illustrated in Fig. 2. These findings demonstrate that the proposed model has significant advantages in supervised learning and is capable of identifying significant differences between normal and abnormal traffic.

5.2 Detection of unlabeled data sets

Table 4 shows the results of the unlabeled data set. CNN and the model proposed in this paper recognized more abnormal samples than the LSTM-based approach. Specifically, MT_MODEL refers to model validation-based data, i.e., unlabeled data, which exists in multi-domain form; MT_RULE is rule-based data containing category imbalanced data to validate the robustness of the model; MT_NEW is based on meta-learning, where new samples are continuously added during the training of the model to validate the scalability of the model; TP naturally ground is the one that is classified as positive samples and positive samples, which discriminates the sample correctly. From Table 4, it is clear that the MT_MODEL indicator of the proposed model is larger than other indicators, which means that most of the traffic that is marked as abnormal. Similarly, the performance of our model for MT_RULE and MT_NEW is also the best, which can be attributed to our application of multi-domain machine learning to effectively solve the heterogeneous problem of distribution. In particular, the use of attention mechanisms allows the model to focus on certain classes of attacks with few samples. Compared with LSTM and Bi-LSTM, the proposed model can find more abnormal traffic. The FP value of the proposed model is the lowest, which illustrates the superiority of the attention-based model.

5.3 Weight distribution of attention

By extracting the attention weight of abnormal traffic, three HTTP traffic entries are obtained that belong

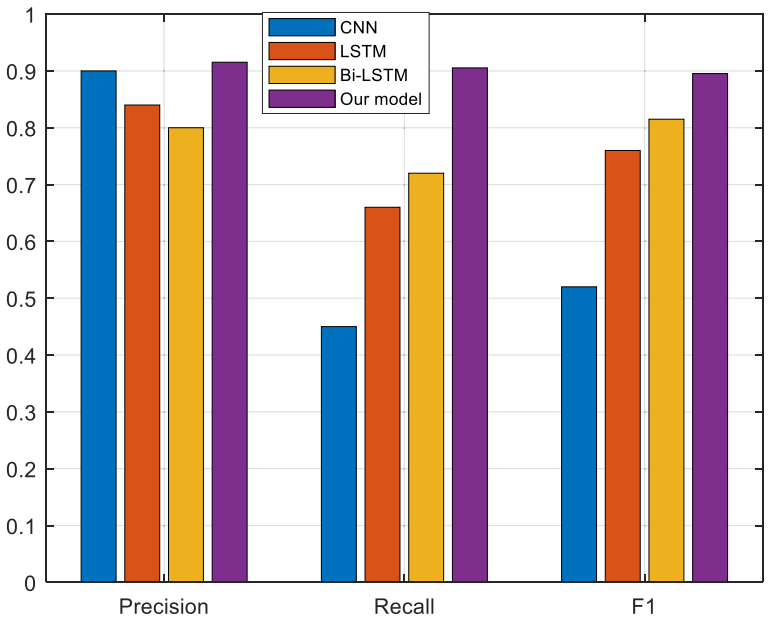

Fig. 2 Results on HTTP-labeled data sets

Table 4 Results on the unlabeled data sets

Model	MT_MODEL	MT_RULE	MT_NEW	FP
CNN	48,270	16,809	4973	33,910
LSTM	38,689	10,276	2527	25,886
Bi-LSTM	40,621	13,302	3835	23,466
Proposed model	55,692	18,917	5974	23,379

to cross-site scripting attacks with similar requests. Figure 3 shows the attention weight distribution. The depth of the color corresponds to the importance of the words in the sentence given by the model. The darker the color, the more important the word, vice versa. Obviously, the valid words detected by the model are "textarea", "<", ">", "script", "alert", and "adminDir," which are consistent with the ground truth.

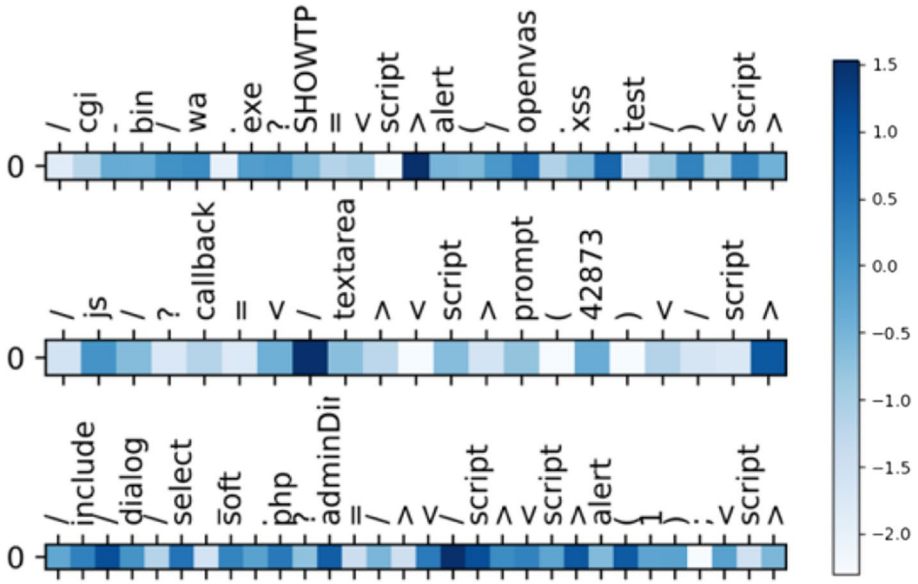

Fig. 3 Weight distribution of attention

5.4 Malware detection

Figure 4 illustrates the precision and recall rates of the model in malware detection. Compared to PCNN, STL, and the proposed model, they have achieved higher precision and recall rates. The precision and recall of the proposed model surpass those of STL for Neris and Emotet, which exhibit clear patterns. Avzhan's ambiguity significantly reduces the overall performance of the proposed model to 0.93. Although Tinba's dimensionality is high, its pattern is also clear. STL and the proposed model can achieve high precision. The proposed model leverages the information of the integrated module, allowing tasks in different domains to share lower-level parameters. The individual level learns specific knowledge, while the public level learns shared knowledge across different domains.

Table 5 presents a comparison of the performance of various models in terms of precision and recall. The "average" column indicates the average performance of each model as shown in Figs. 4 and 5. The results demonstrate that the proposed model exhibits superior stability, with minimal reduction in precision and significant improvement in recall rate.

5.5 Identification of VPN encapsulation

The results of the performance comparison between the proposed model and the decision tree in Table 6 indicate a significant improvement in precision and recall rate for both VPN and non-VPN data, with an average improvement of 9.25%. This demonstrates that the proposed model is more effective in detecting network attacks than the decision tree, which may struggle to distinguish between normal and abnormal traffic due to the often-disguised nature of network attacks. Overall, these findings suggest that the proposed approach is a promising method for improving network security and mitigating the risks associated with cyber threats.

5.6 Trojan Horse classification

Table 7 presents the overall performance of different models on the CTU-13 data set. The proposed model outperforms PCNN and HDM in terms of precision and recall rate, while also saving almost one-third of the training time compared to PCNN and HDCM. This is due to the ability of the proposed model to learn three domain tasks simultaneously through parameter sharing among domains.

Table 5 Performance of models

	Precision (%)			Recall (%)		
	PCNN	STL	Proposed model	PCNN	STL	Proposed model
Average	94.6	96.7	97.9	94.9	96.6	98.0
Optimal	94.7	96.9	98.2	95.2	95.3	98.6
Difference	0.1	0.2	0.3	0.3	0.7	0.6

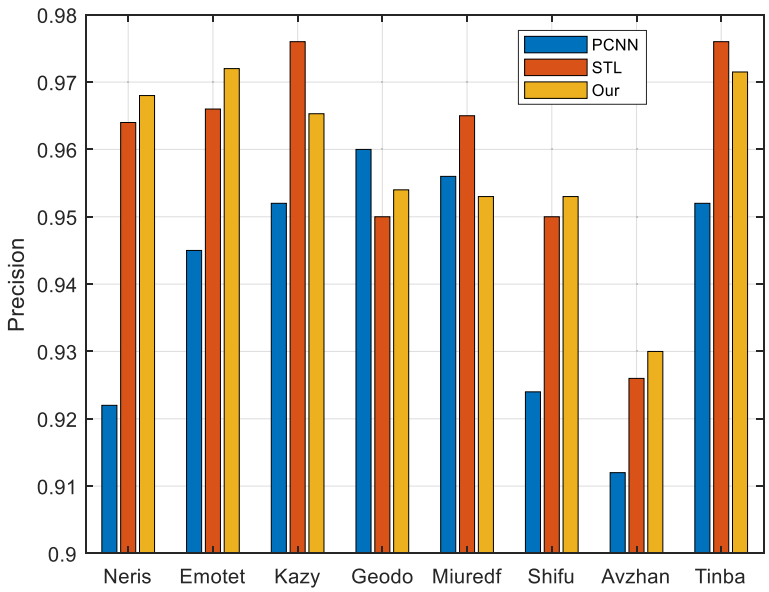

Fig. 4 The precision of each malware detection

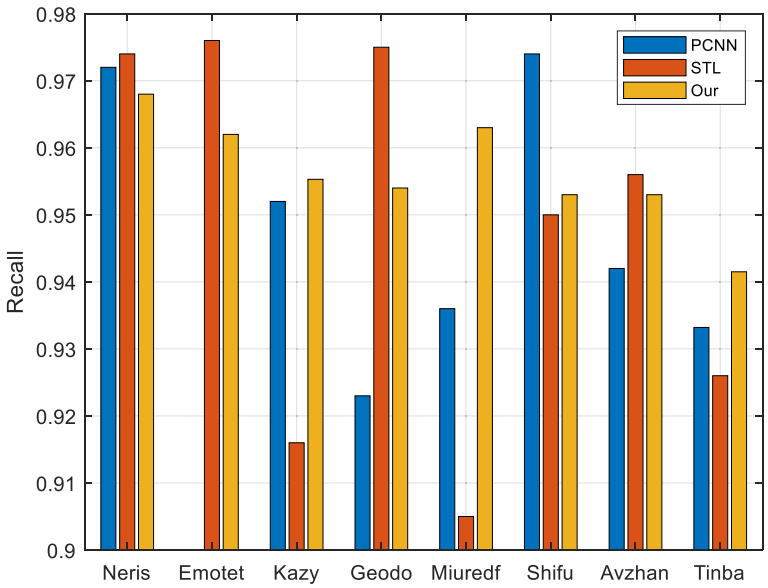

Fig. 5 The recall of each malware detection

Table 6 Comparison between the decision tree and the proposed model

Models	VPN (%)		Non-VPN (%)	
	Precision	Recall	Precision	Recall
Decision tree	89.0	92	91.6	87.8
Proposed model	99.3	99.5	99.7	98.6
Performance improvement	+10.9	+6.8	+9.3	+10.0

The confidence interval shown in Table 8 was obtained using paired sampling t test with a 90% confidence level. The subscripts 1, 2, and 3 refer to PCNN, HDM, and the proposed model, respectively. The degrees of freedom were 24 (the abnormal category was 9, and there were three models). The results demonstrate that the proposed model outperforms both PCNN and HDM models in terms of recall rate, and is better than HDM in precision. Furthermore, the results show that the proposed model can handle three different domain tasks simultaneously without compromising performance in any single domain.

Table 7 Overall performance of the models

Measures	PCNN	HDCM	Proposed model
Precision (%)	97.5	94.8	99.4
Recall (%)	94.7	94.7	98.0
Training time (s)	25.42	21.75	9.19

Table 8 Confidence interval

	$\mu_1 - \mu_3$	$\mu_2 - \mu_3$
Precision (%)	(−3.561, 1.316)	(−6.711, −1.985)
Recall (%)	(−6.671, −0.482)	(−6.735, −0.281)

5.7 KDDcupmulti detection

Table 9compares the performance of the PCNN model, which is a lightGBM-based network traffic anomaly detection model that utilizes dataoptimization and Focal loss optimization, with our model, which onlyutilizes data optimization. The recall values for all four attacktypes have improved, with Normal decreasing by only 2% and Dos, U2R,R2L, and Probe increasing by 5%, 2%, 8%, and 6% respectively. Amongthese improvements, the recall value for the R2L attack type has beenthe most significant. Ourmodel is a multi-classifier network traffic anomaly detection modelthat uses data balancing and feature selection after optimization.Compared to the PCNN model,

Table 9 Comparison of Recall values of PCNN and our model on KDDcup99

	Recall	
	PCNN	Our model
Normal	0.94	0.92
Dos	0.81	0.87
U2R	0.13	0.15
R2L	0.29	0.37
Probe	0.87	0.93

Table 10 DetectionEffectiveness of Different Types of Malware

Category	Number of families	Number of samples	F1 score
Adware	48	47,210	0.8595
Backdoor	11	1,538	0.9627
File Infector	5	669	0.8963
No Category	-	2,296	0.9756
PUA	8	2,051	0.9786
Ransomware	8	6,202	0.9775
Riskware	21	97,349	0.9555
Scareware	3	1,556	0.9788
Trojan	5	2351	0.7868

which is solely based on CNN without anyoptimization, our model has achieved an average recall improvement of14% across all attack types.It is worth noting that the recall value for U2Rand R2L is quite low, indicating that these types of malware may beless common or more difficult to detect than other types. The recallvalue for Probe is also lower than the other types, which suggeststhat our model may need further improvement in detecting this type ofmalware.

5.8 Multi-domaindetection on CCCS-CIC-AndMal2020 data set

Androidmalware is one of the most serious threats on the internet, and hasseen an unprecedented surge in recent years. This presents a publicchallenge to cybersecurity experts. There are many techniquesavailable for identifying and classifying Android malware based onmachine learning, but recently, multi-domain machine learning hasemerged as a prominent classification method for such samples.Table10presents the results of a malware detection experiment, with eachcategory of malware listed along with the number of families andsamples detected, as well as the F1 score.Itappears that the most common types of malware detected were adware,backdoor, file infector, riskware,ransomware, and Trojan. These categories accounted for a significantportion of the total number of families and samples detected.Itis worth noting that there are some differences in the F1 scoresbetween different categories. For example, the F1 score forransomware was relatively high (0.9775), while the F1 score fortrojan was relatively low (0.7868). This suggests that differenttypes of malware may require different approaches for detection andclassification. As can be seen from Table 10,the F1scorefor each category of Android malware detection in this paper is veryhigh, averaging 0.92, meeting the requirements of real-worldscenarios and providing strong protection for Android phones.Moreover, the efficiency and accuracy of the multi-domain detectionsystem have been greatly improved.

5.9 Model convergence

The convergence validation of the proposed model is shown in Fig. 6. According to Fig. 6, the weight search space of the neural network is smaller than that of a single structure LSTM. The network structure and the number of neurons in the combination of attention mechanism

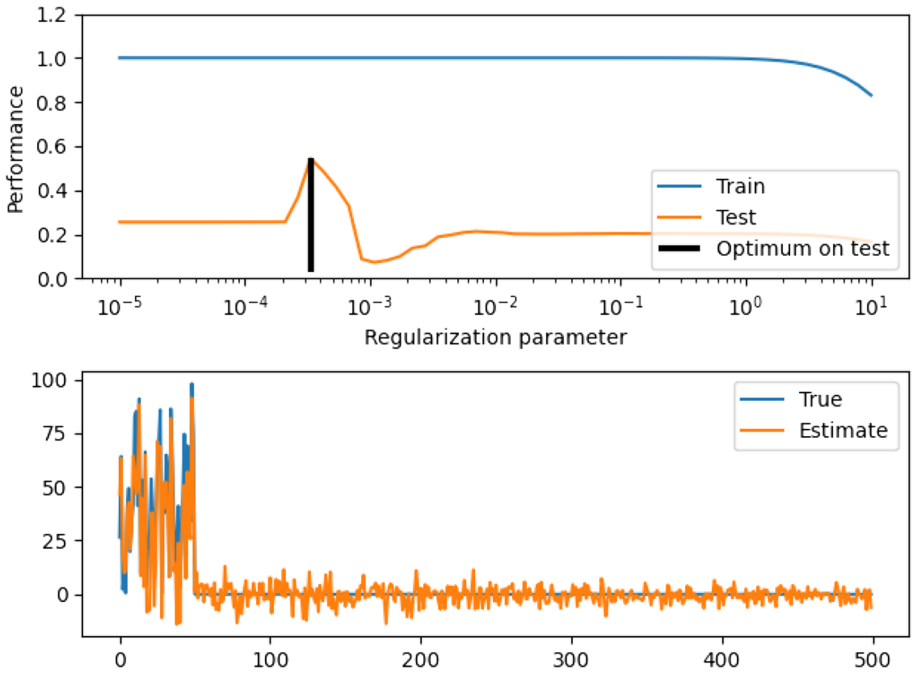

Fig. 6 Convergence of the proposed model

and multi-domain machine learning affect the convergence of the model. The attention mechanism corrected by weighted summation is independent of the output of the multi-domain learning [32]. Auto-regressive information is incorporated in the model, which improves the accuracy of the proposed algorithm. Besides, the lower panel of Fig. 6 also shows that the objective function converges rapidly and tends to smooth out (i.e., the convergence speed of weights degrades shapely from 50 to around 0) during both the testing and training processes. Because the feedback information of the model structure restricts the weight search space of the LSTM-based neural network, it is easier for the proposed algorithm to obtain the optimal value and fast convergence speed.

6 Conclusion

In this study, we propose a multi-domain machine learning model for network attack anomaly detection by combining Bi-LSTM and attention mechanism. Our approach has been tested on several public large-scale traffic data sets, and the results demonstrate that integrating multiple modules does not lead to additional overfitting. The proposed model is able to accurately detect different types of network anomalies, including HTTP traffic anomaly detection, malware detection, VPN encapsulation identification, and Trojan horse classification. One key advantage of our approach is its ability to adapt to different scenarios and optimize the learning system accordingly. In future work, we plan to explore the internal structure of various neural networks further in order to achieve even better performance. This could involve analyzing the strengths and weaknesses of different models, as well as experimenting with new training techniques and algorithms. Overall, our research demonstrates the potential of using advanced machine learning techniques for network security. By combining multiple modules and optimizing the learning system, we can develop more effective methods for detecting and preventing network attacks. As such, our work has important implications for both individuals and organizations looking to protect their online assets from malicious actors.

Abbreviations

Bi-LSTM	Bi-directional long-short-term memory
DBN	Deep belief network
TF-IDF	Term frequency-inverse document frequency
XSS	Cross-site scripting
CNN	Deep feedforward artificial neural networks
RNN	Recurrent neural networks
PCCN	Parallel cross-convolutional neural network
HDCM	Hybrid of FNN (feed-forward neural networks) and CNN
STL	Single task learning of the basic layer of the proposed model

Authors' contributions

XW provided the original idea, completed the experiments, and formed the document. JL completed the experiments and visualized the results. CZ implemented the algorithms and reviewed and revised this paper.

Funding

This work was supported by Research on the Construction of Practical Teaching System Based on "Five Talents Training Standards" – Taking Big Data Technology Application Specialty As an Example (YJJG2021007) and Research on the Dilemma of Building Morality and Cultivating Talents in Higher Vocational School-Enterprise Cooperation—Taking Big Data Technology as an Example (XJ2223).

Availability of data and materials

All the data are available publicly.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 20 March 2022 Accepted: 23 May 2023
Published online: 26 June 2023

References

1. L. Deng, G. Xie, H. Liu, Y. Han, R. Li, K. Li, A survey of real-time ethernet modeling and design methodologies: from AVB to TSN. ACM Comput. Surv. (CSUR) **55**(2), 1–36 (2022)
2. C. Zhang, X. Costa-Pérez, P. Patras, Tiki-Taka: attacking and defending deep learning-based intrusion detection systems, in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*. (2020), pp.27–39
3. B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: identifying and mitigating backdoor attacks in neural networks, in *2019 IEEE Symposium on Security and Privacy (SP)*. (IEEE, San Francisco, 2019), pp. 707–723
4. Z. Shi, J. Li, C. Wu et al., DeepWindow: an efficient method for online network traffic anomaly detection, in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. (IEEE, Zhangjiajie, 2019), pp. 2403–2408
5. N. Moustafa, B. Turnbull, K.K.R. Choo, An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet Things J. **6**(3), 4815–4830 (2018)
6. N. Moustafa, K.K.R. Choo, I. Radwan et al., Outlier Dirichlet mixture mechanism: adversarial statistical learning for anomaly detection in the fog. IEEE Trans. Inf. Forensics Secur. **14**(8), 1975–1987 (2019)
7. B. Mbarek, M. Ge, T. Pitner, Enhanced network intrusion detection system protocol for internet of things, in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. (2020), pp.1156–1163
8. A. Juvonen, T. Sipola, T. Hämäläinen, Online anomaly detection using dimensionality reduction techniques for HTTP log analysis. Comput. Netw. **91**, 46–56 (2015)
9. Z. Zhang, Q. He, J. Gao et al., A deep learning approach for detecting traffic accidents from social media data. Trans. Res. Part C Emerg. Technol. **86**, 580–596 (2018)
10. H. Zhang, I. Goodfellow, D. Metaxas et al., Self-attention generative adversarial networks, in *International conference on machine learning*. (PMLR, Long Beach, 2019), pp. 7354–7363
11. M. Joshi, M. Dredze, W. Cohen et al., Multi-domain learning: when do domains matter?, in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. (2012), pp.1302–1312
12. H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2016), pp.4293–4302
13. M.A. Ferrag, L. Maglaras, S. Moschoyiannis et al., Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J. Inform. Secur. Appl. **50**, 1–19 (2020)
14. G. Liu, J. Guo, Bidirectional LSTM with attention mechanism and convolutional layer for text classification. Neurocomputing **337**, 325–338 (2019)

15. Y. Wang, J. An, W. Huang, Using CNN-based representation learning method for malicious traffic identification, in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*. (IEEE, Singapore, 2018), pp. 400–404
16. D. Kwon, K. Natarajan, S.C. Suh, H. Kim, J. Kim, An empirical study on network anomaly detection using convolutional neural networks, in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. (IEEE, Vienna, 2018), pp. 1595–1598
17. Y. Yan, L. Qi, J. Wang, Y. Lin, L. Chen, A network intrusion detection method based on stacked autoencoder and LSTM, in *ICC 2020–2020 IEEE International Conference on Communications (ICC)*. (IEEE, The Convention Centre Dublin, 2020), pp. 1–6
18. Z. Wu, J. Wang, L. Hu, Z. Zhang, H. Wu, A network intrusion detection method based on semantic re-encoding and deep learning. J. Netw. Comput. Appl. **164**, 102688 (2022)
19. J. Zhang, Y. Ling, X. Fu, X. Yang, G. Xiong, R. Zhang, Model of the intrusion detection system based on the integration of spatial-temporal features. Comput. Secur. **89**, 101681 (2020)
20. E. Mushtaq, A. Zameer, M. Umer, A.A. Abbasi, A two-stage intrusion detection system with auto-encoder and LSTMs. Appl. Soft Comput. **121**, 108768 (2022)
21. P. Jeatrakul, K.W. Wong, C.C. Fung, Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm, in *Neural information processing. Models and applications: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part II 17*. (Springer Berlin Heidelberg, 2010), pp.152–159
22. B. Yan, G. Han, LA-GRU: building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural network. Secur. Commun. Netw. **2018**, 13 (2018). https://doi.org/10.1155/2018/6026878. (Article ID 6026878)
23. N. Gupta, V. Jindal, P. Bedi, LIO-IDS: handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system. Comput. Netw. **192**, 108076 (2021)
24. I. Yahav, O. Shehory, D. Schwartz, Comments mining with TF-IDF: the inherent bias and its removal. IEEE Trans. Knowl. Data. Eng. **31**(3), 437–450 (2018)
25. P. Bedi, N. Gupta, V. Jindal, Siam-IDS: handling class imbalance problem in intrusion detection systems using siamese neural network. Procedia. Comput. Sci. **171**, 780–789 (2020)
26. T. Bai, J. Zhao, J. Zhu, S. Han, J. Chen, B. Li, A. Kot, Ai-gan: attack-inspired generation of adversarial examples, in *2021 IEEE International Conference on Image Processing (ICIP)*. (IEEE, Anchorage, 2021), pp. 2543–2547
27. F. Ma, R. Chitta, J. Zhou et al., Dipole: diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks, in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. (Long Beach, 2017), pp. 1903–1911
28. S. Shamshirband, A.T. Chronopoulos, A new malware detection system using a high performance-ELM method, in *Proceedings of the 23rd international database applications & engineering symposium*. (2019), pp.1–10
29. S. Soheily-Khah, P.F. Marteau, N. Béchet, Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: a case study on the ISCX dataset, in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. (IEEE, South Padre Island, 2018), pp. 219–226
30. Y. Zhang, X. Chen, D. Guo et al., PCCN: parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows. IEEE Access **7**, 119904–119916 (2019)
31. H. Huang, H. Deng, Y. Sheng et al., Accelerating convolutional neural network-based malware traffic detection through ant-colony clustering. J. Intell. Fuzzy. Syst. **37**(1), 409–423 (2019)
32. P. An, Z. Wang, C. Zhang, Ensemble unsupervised autoencoders and Gaussian mixture model for cyberattack detection. Inf. Process Manag. **59**(2), 102844 (2022)

Publisher's Note