IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 4, 2017, accepted November 4, 2017, date of publication December 11, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2780250

HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection

WEI WANG[1], YIQIANG SHENG[2], JINLIN WANG[2], XUEWEN ZENG[2], XIAOZHOU YE[2], YONGZHONG HUANG[3], AND MING ZHU[1]

[1]Department of Automation, University of Science and Technology of China, Hefei 230026, China
[2]National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
[3]Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Wei Wang (ww8137@mail.ustc.edu.cn)

This work was supported by the Pioneer Program of Institute of Acoustics, Chinese Academy of Sciences, under Grant Y654101601.

ABSTRACT The development of an anomaly-based intrusion detection system (IDS) is a primary research direction in the field of intrusion detection. An IDS learns normal and anomalous behavior by analyzing network traffic and can detect unknown and new attacks. However, the performance of an IDS is highly dependent on feature design, and designing a feature set that can accurately characterize network traffic is still an ongoing research issue. Anomaly-based IDSs also have the problem of a high false alarm rate (FAR), which seriously restricts their practical applications. In this paper, we propose a novel IDS called the hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS), which first learns the low-level spatial features of network traffic using deep convolutional neural networks (CNNs) and then learns high-level temporal features using long short-term memory networks. The entire process of feature learning is completed by the deep neural networks automatically; no feature engineering techniques are required. The automatically learned traffic features effectively reduce the FAR. The standard DARPA1998 and ISCX2012 data sets are used to evaluate the performance of the proposed system. The experimental results show that the HAST-IDS outperforms other published approaches in terms of accuracy, detection rate, and FAR, which successfully demonstrates its effectiveness in both feature learning and FAR reduction.

INDEX TERMS Network intrusion detection, deep neural networks, representation learning.

I. INTRODUCTION

Cyberspace security has recently gained increasing attention. Creating effective defenses against various types of network attacks and ensuring the safety of network equipment and information security has become a highly considered problem. Network intrusion detection systems (IDSs) identify malicious attack behaviors by analyzing the network traffic of key nodes of a network and have become an important part of the network security protection architecture.

The anomaly-based detection method, which is a primary research direction in the field of intrusion detection, learns normal and anomalous behaviors by analyzing network traffic and can detect unknown and new attacks [1]. However, its performance is highly dependent on feature design, and designing a feature set that can accurately characterize network traffic is still an ongoing research issue [2]. Anomaly-based IDS also have a high false alarm rate (FAR), which seriously restricts its practical application [3].

To solve the abovementioned problems, using the representation learning approach, we propose a hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS) that automatically learns network traffic features. It first learns the spatial features of network traffic using deep convolutional neural networks (CNNs) and then learns the temporal features using long short-term memory (LSTM) networks, which are a special type of recurrent neural networks (RNNs). The high-level temporal features are based on the low-level spatial features. The entire feature-learning process is conducted automatically: no feature engineering techniques are required. We expect that the automatically learned hierarchical spatial-temporal features are better at characterizing network traffic behaviors than are manually designed features and that they can effectively improve the intrusion detection capability. The experimental results successfully demonstrate the effectiveness of the proposed method for both feature learning and FAR reduction.

The rest of this paper is organized as follows. Section II describes the related work. Section III describes the design and implementation of the proposed method. Section IV mainly covers the evaluation methodology and experimental results, and Section V presents conclusions and future work.

II. RELATED WORK

A. INTRUSION DETECTION TECHNIQUES

IDSs can be classified as either signature-based or anomaly-based detection. Signature-based detection, also called misuse detection, analyzes known attacks to extract their discriminating characteristics and patterns, called signatures. These signatures are compared against the new traffic to detect intrusions. The advantages of signature-based detection are that it has a high detection rate and a low FAR for known attacks, while its disadvantage is that it cannot detect any unknown and new (0-day) attacks. Anomaly-based detection, also called behavior-based detection, mainly uses a machine learning-based method. In this approach, some network traffic features are designed first; then, a model is built based on those features using supervised or unsupervised learning approaches. The model can identify both normal and anomalous traffic patterns. Its advantage is that it can detect unknown and new attacks; thus, it has attracted increasing interest in research and industrial circles.

However, in practical applications, some problems still exist for anomaly-based intrusion detection methods. The first problem is the difficulty of designing representative traffic features. The detection effect of this method is highly dependent on the design of the traffic features used in training. The detection effects often vary widely when different feature sets of network traffic are applied. No standard guiding principle currently exists for the design of a feature set that accurately characterizes network traffic. The second serious problem of the anomaly-based intrusion detection method is its high FAR, which is a major obstacle to its practical application [3].

The representation learning approach [4] is a promising method for solving both these problems. Representation learning, also called feature learning, is a technique that allows a system to automatically extract features from raw data. Its biggest advantage is that it replaces manual feature engineering and can directly learn the best features from raw data. Deep neural networks have been the most successful technique of representation learning and have achieved remarkable results in the fields of computer vision and natural language processing. In the field of network intrusion detection, some research results obtained using the representation learning approach have recently emerged. For example, Ma *et al.* [5] applied deep neural networks to detect intrusion behaviors using the KDD99 dataset. Niyaz *et al.* [6] studied the intrusion detection method on the NSL-KDD dataset using deep belief networks. The common ground of these research methods is that their models learn features from manually designed traffic features. However, applications in the fields of computer vision and natural language processing have shown that the biggest advantage of deep neural networks lies is their ability to learn features directly from raw data [7]. The abovementioned research methods used deep neural networks based on manually designed features without taking full advantage of the deep neural networks.

The literature reveals that raw network traffic data have not been used to learn features. This revelation motivates us to develop a process for learning features directly from raw network traffic data using deep neural networks to obtain a better traffic feature set and develop a more efficient IDS. Additionally, Eesa *et al.* [8] proved that the detection rate can be increased and the FAR decreased by using a better traffic feature set. We also hope to reduce the FAR by using an automatically learned traffic feature set.

B. DEEP NEURAL NETWORKS

CNNs and RNNs are the two most widely used deep neural network models; they are capable of learning effective spatial and temporal features, respectively. In common neural networks, every neural node of every hidden layer sums the weighted values coming from the previous layer, applies a nonlinear transform, and transfers the resulting values to the next layer. The output value of the last layer can be regarded as the representation or feature learned by the neural networks from the input data. CNNs, which improve upon the architecture of the common neural networks, benefit from the following: sparse connectivity, shared weights and pooling. CNNs are capable of learning spatial features and have already yielded impressive achievements in many computer vision tasks, such as image classification [9]. RNNs add a self-connected weighted value as the memory unit for every neural node based on the architecture of common neural networks, and they can memorize the previous state of the neural networks. Long short-term memory (LSTM) networks further add a component called the forget gate, and LSTMs can effectively learn temporal features from a long sequence. LSTM networks have already been shown to perform well for many natural language processing tasks, such as machine translation [10].

Spatial and temporal features are two commonly used types of traffic features in the field of intrusion detection. When using spatial features, network traffic is first transformed into traffic images; then, the image classification method based on geometric features is used to classify the traffic images, which also indirectly achieves the goal of identifying the malware traffic. This approach is relatively new, but many recent research results demonstrate its great potential. For example, Tan *et al.* [11] applied a widely used dissimilarity measure in the computer vision domain, namely, the earth mover's distance (EMD), to detect denial of service (DoS) traffic and achieved a good effect. When using temporal features, the temporal features of a network traffic flow are extracted and can be used to detect intrusion behaviors via the time series analysis method. This approach was developed early and has been adopted by many researchers. For example, Ariu *et al.* [12] developed an effective IDS using the hidden

Markov model method based on the temporal relations of traffic payload bytes.

Over the past two years, a few studies have used CNNs or RNNs to perform intrusion detection tasks based on spatial and temporal features. For example, Wang *et al.* [13] used a CNN to learn the spatial features of network traffic and achieved malware traffic classification using the image classification method. Torres *et al.* [14] first transformed network traffic features into a sequence of characters and then used RNNs to learn their temporal features, which were further applied to detect malware traffic. The common point of these research methods is that they used CNNs or RNNs alone and learned a single type of traffic feature.

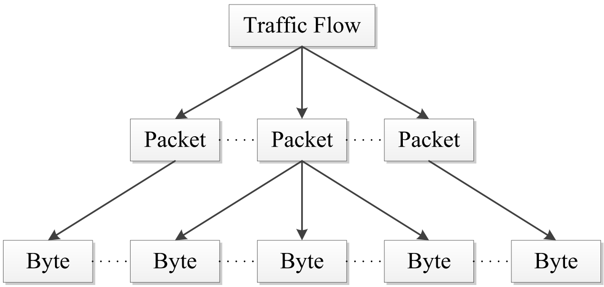


FIGURE 1. Hierarchical structure of network traffic.

Network traffic has an obvious hierarchical structure as illustrated in Figure 1, where the bottom row shows a sequence of traffic bytes. Based on the format of specific network protocols, multiple traffic bytes are combined to form a network packet, and multiple network packets communicated between two sides are further combined to form a network flow. Notably, these traffic bytes, network packets and network flows are similar to the characters, sentences and paragraphs in the field of natural language processing. Moreover, the task of classifying a network flow as either normal or malware is very similar to classifying a paragraph as either positive or negative, which is a common task in the field of natural language processing, namely, sentiment analysis. In some recent studies on sentiment analysis, deep neural networks were used to learn the hierarchical features of natural language and achieved good results [15]–[17]. Those studies motivated us to use deep neural networks to learn the hierarchical features of network traffic and further perform the intrusion detection task.

However, no studies that combine the use of CNNs and RNNs to detect network intrusion exist in the literature. To take full advantage of these two types of deep neural networks, we use both CNNs and RNNs to learn the spatial-temporal features of raw network traffic data to develop a more effective IDS.

III. HAST-IDS

A. HAST-IDS OVERVIEW

The goals of the HAST-IDS are to automatically learn the spatial-temporal features of raw network traffic data using deep neural networks and to improve the effectiveness of the IDS. The basic design concept is as follows. At the network packet level, every network packet is transformed into a two-dimensional image, the internal spatial features of which are learned by a CNN. At the network flow level, the temporal features of a sequence of network packets of a network flow are further learned by an RNN. Finally, the resulting spatial-temporal traffic features are used to classify the traffic as normal or malware.

Two implementation schemes are used for the HAST-IDS, as shown in Figure 2. HAST-I uses a CNN and learns only spatial features, while HAST-II uses both CNNs and RNNs and learns spatial-temporal features. Each scheme has different application scenarios for different types of network traffic, which will be discussed in Section IV.

The various stages of the HAST-IDS are described below.

1) PREPROCESSING

In this stage, the input raw network traffic data are transformed into the two-dimensional images required by the CNN. The basic traffic units for intrusion detection are network flows; thus, the input raw traffic data must be split into multiple network flows. Each network flow contains multiple network packets communicated between two endpoints. One-hot encoding (OHE) is used as the transformation method. In HAST-I, the first n traffic bytes of the network flow are transformed. If the OHE vector is m-dimensional, then the entire network flow can be transformed into an m*n two-dimensional image. For HAST-II, similar preprocessing is required; however, every network packet must be transformed individually. If r packets exist in a flow and if the first q bytes of every packet are transformed and the OHE vector is p-dimensional, then the entire network flow can be transformed into r different p*q two-dimensional images.

2) CROSS-VALIDATION

The k-fold cross-validation technique is used for performance evaluation. In this technique, a dataset is randomly divided into k equal parts. In each iteration, one part is selected as the validating dataset, while all the other k-1 parts are treated as the training dataset. In our experiments, k was set to 10 because of the resulting low bias, low variance, low overfitting and good error estimate [18].

3) SPATIAL FEATURE LEARNING

CNNs are used to learn the spatial features of the two-dimensional traffic images. In HAST-I, the spatial features of the entire flow image are learned from a single m*n image, and the output is a single flow vector. In HAST-II, the spatial features of every p*q packet image are learned individually, and the output is r packet vectors.

4) TEMPORAL FEATURE LEARNING

An LSTM is used to learn the temporal features of multiple traffic vectors. In HAST-II, the LSTM further learns the temporal relations among the r packet vectors. The output

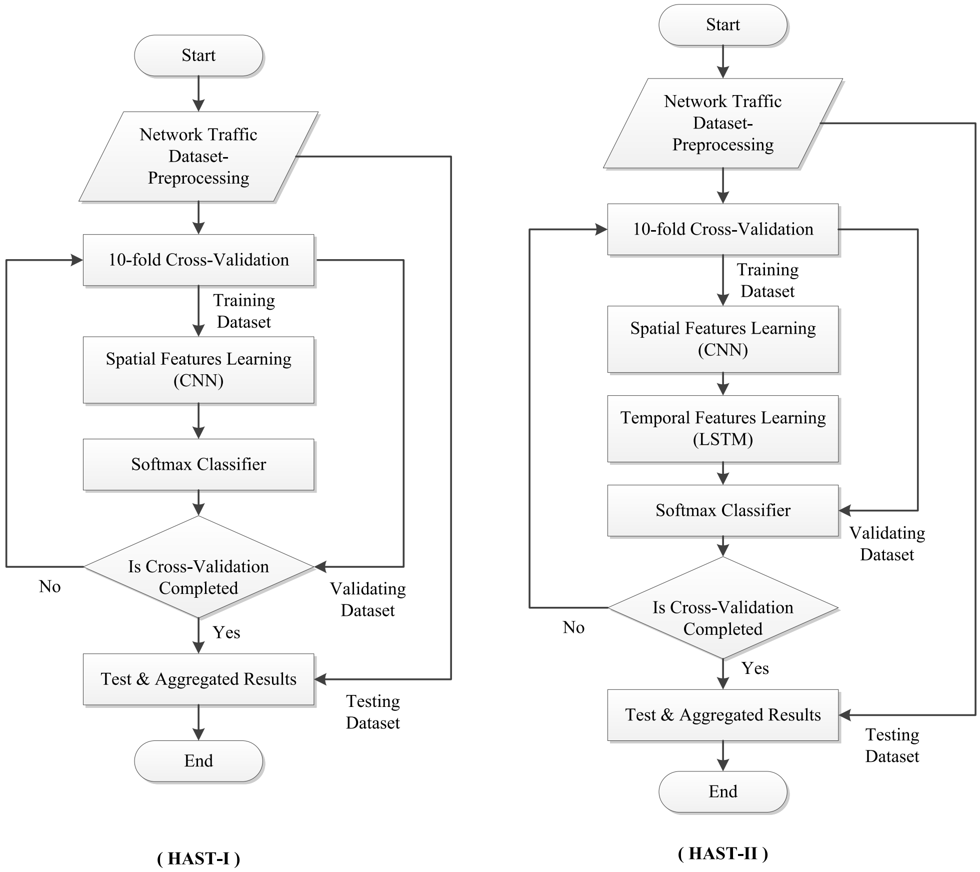

Start
Network Traffic Dataset-Preprocessing
10-fold Cross-Validation
Training Dataset
Spatial Features Learning (CNN)
Softmax Classifier
Is Cross-Validation Completed
No
Yes
Validating Dataset
Test & Aggregated Results
Testing Dataset
End
(HAST-I)
Start
Network Traffic Dataset-Preprocessing
10-fold Cross-Validation
Training Dataset
Spatial Features Learning (CNN)
Temporal Features Learning (LSTM)
Softmax Classifier
Validating Dataset
Is Cross-Validation Completed
No
Yes
Test & Aggregated Results
Testing Dataset
End
(HAST-II)

FIGURE 2. Workflow of HAST-IDS.

is a single flow vector that represents the spatial-temporal features of the network flow.

5) SOFTMAX CLASSIFIER

The softmax classifier is used to determine whether the input traffic is normal or malware based on the flow vector. Softmax is a commonly used multi-class classification method in the field of machine learning.

6) TEST AND RESULT AGGREGATION

The fine-tuned model is tested using the test dataset. The results of each experiment are collected and analyzed.

B. LEARNING SPATIAL FEATURES WITH CNNS

CNNs are used to learn the spatial features from the two-dimensional image of the input network traffic bytes, as shown in Figure 3. In HAST-I, the CNN is applied to the entire network flow. The first n traffic bytes of the entire flow are transformed into a single m*n flow image via OHE. The image is further processed via convolution and pooling. The final output is a flow vector that represents the features of the entire flow. In HAST-II, the CNN is applied to every network packet. The first q bytes of every packet are transformed into a p*q packet image via OHE, and each image is further processed via convolution and pooling. The final output consists of multiple packet vectors that represent the features of the individual network packets. When implemented, two convolution filters with different sizes are used to output two different flow/packet vectors, which are concatenated together as the final vector. The application of CNNs in the field of computer vision has shown that this method can obtain better spatial features. Algorithm 1 presents the details of the spatial feature learning stage.

The key components used in Algorithm 1 are as follows.

1) ONE-HOT ENCODING

Let $x_i \in \mathbb{R}$ be the k-dimensional vector corresponding to the i-th traffic byte in a packet or flow. A packet or flow of length n can be encoded according to the following formula, where $\oplus$ is the concatenation operator. In general, $x_{i:i+j}$ denotes the concatenation of traffic bytes $x_i, x_{i+1}, \ldots, x_{i+j}$.

$$x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n \qquad (1)$$

2) CONVOLUTION OPERATION

A convolution operation involves a filter $w \in \mathbb{R}$, which is applied to a window of h traffic bytes to produce a new feature. For example, a feature c_i is generated using this formula, where $b \in \mathbb{R}$. is a bias term, and f denotes ReLUs [19]

$$c_i = f(w \cdot x_{i:i+h-1} + b) \qquad (2)$$

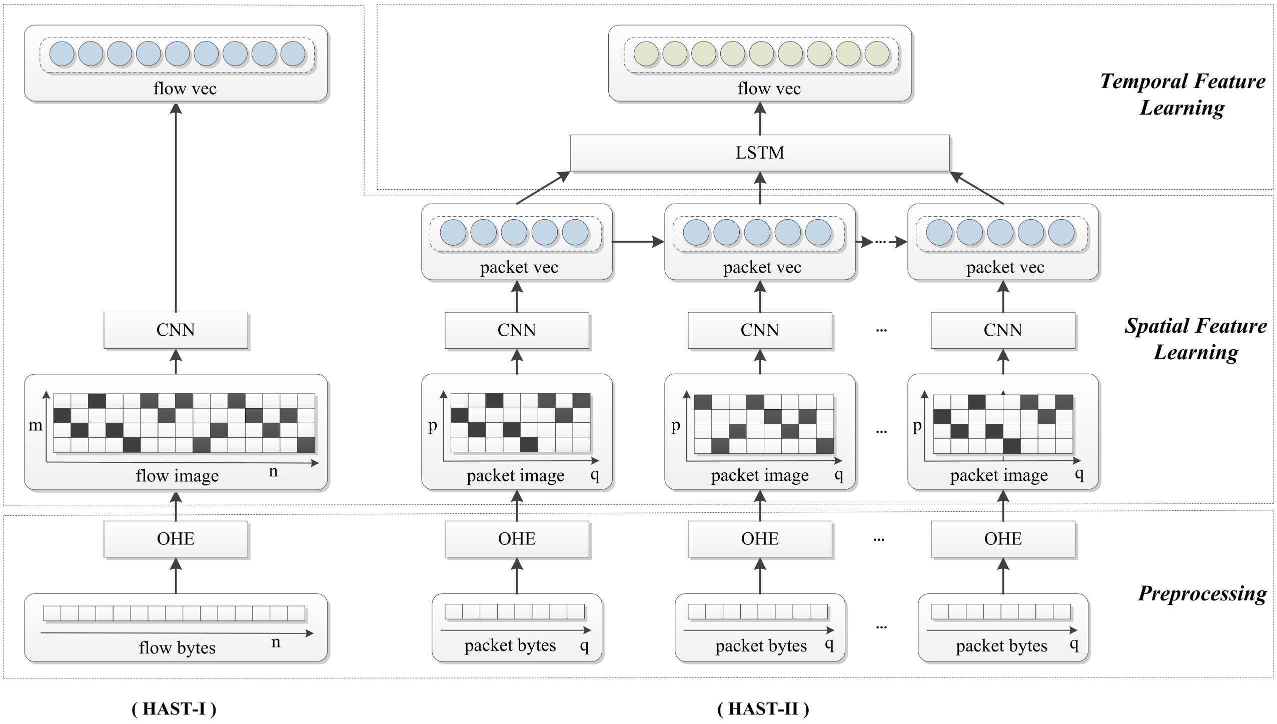


FIGURE 3. **Feature learning process of HAST-IDS.**

3) FEATURE MAPPING

A convolution filter is applied to each possible window $\{x_{1:h}, x_{2:h+1}, \cdots x_{n-h+1:n}\}$ to produce a feature map with $c \in \mathbb{R}$.

$$c = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

4) POOLING OPERATION

A max-over-time pooling operation is then applied to the feature map and takes the maximum value as the final feature.

$$\hat{c} = \max\{c\}. \tag{4}$$

C. LEARNING TEMPORAL FEATURES WITH LSTM

An LSTM [20] is used to learn the temporal features based on the sequence of network packet vectors, as shown in Figure 3. In HAST-II, a bidirectional LSTM is used to scan the sequence both from beginning to end and in reverse. The application of LSTM in the field of natural language processing has shown that bidirectional scanning can yield more accurate features [10]. Algorithm 2 presents the details of the temporal feature learning stage.

Compared to common neural networks and the common RNN, an LSTM offers several key improvements for each neural node as described below.

1) FORGET GATE

The forget gate provides a forgetting coefficient by looking at the input layer x_t and previous hidden layer h_{t-1} for cell state C_{t-1}. The coefficient ranges from 0 to 1 and controls the information between C_{t-1} and C_t.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

2) INPUT NODE

This unit also considers the input layer x_t and the previous hidden layer h_{t-1}. Typically, a *tanh* layer is used to process the summed weighted input.

$$g_t = tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{6}$$

3) INPUT GATE

The input gate decides which values should be updated in C_{t-1}, and its output multiplies the value of the input node to generate a new candidate for C_t.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7}$$

4) INTERNAL STATE

The memory unit combines the computation results mentioned above.

$$C_t = f_t * C_{t-1} + i_t * g_t \tag{8}$$

5) OUTPUT GATE

The hidden layer h_t is produced based on the internal state C_t and the value of the output gate o_t.

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{9}$$

Algorithm 1 Spatial Feature Learning

```
  Input: Network traffic image: a flow image ($f_1$) in HAST-I or r packet images ($p_1, p_2, \ldots, p_r$) in HAST-II.
  Output: Spatial features of network traffic: a flow vector ($v_{f1}$) or r packet vectors ($v_{p1}, v_{p2}, \ldots, v_{pr}$).
  Step 1: Create CNN model_1
     1. Add 1st convolution layer with $c_1$ filters of size $s_1$, followed by 1st max pooling layer of size $t_1$.
     2. Add 2nd convolution layer with $c_2$ filters of size $s_2$, followed by 2nd max pooling layer of size $t_2$.
     3. Add 1st dense layer, the output of which is a temp vector $V_{temp1}$.
  Step 2: Create CNN model_2
     4. Add 3rd convolution layer with $c_3$ filters of size $s_3$, followed by 3rd max pooling layer of size $t_3$.
     5. Add 4th convolution layer with $c_4$ filters of size $s_4$, followed by 4th max pooling layer of size $t_4$.
     6. Add 2nd dense layer, the output of which is a temp vector $V_{temp2}$.
  Step 3: Concatenate two temp vectors
     7. $V_{temp} = V_{temp1} + V_{temp2}$
  Step 4: Train and validate model
     8. while early termination condition is not met, do
        while training dataset is not empty, do
          Prepare the mini-batch dataset as the model input.
          Compute the categorical cross-entropy loss function $H(p, q) = -\sum_x p(x) log(q(x))$, p = true_dist and q = predict_dist.
          Update the weights and biases using the RMSprop gradient descent optimization algorithm.
        end
        Validate model using the validating dataset.
      end
  Step 5: Test model
     9. Test the fine-tuned model using the test dataset.
    10. return the $V_{temp}$ of every network traffic image in the test dataset.
```

Algorithm 2 Temporal Feature Learning

```
    Input: r packet vectors of network traffic ($v_{p1}, v_{p2}, \ldots, v_{pr}$).
    Output: spatial-temporal features of network traffic: a flow vector ($v_f$).
  Step 1: Create LSTM model
    1. Add 1st LSTM layer of $l_1$ units, with dropout $d_1$ and recurrent dropout $r_1$.
    2. Add 2nd LSTM layer of $l_2$ units, with dropout $d_2$ and recurrent dropout $r_2$.
    3. Add a dense layer whose output is a flow vector $v_f$.
  Step 2: Train and validate model
    4. Train and validate the model as described in Algorithm 1.
  Step 3: Test model
    5. Test the fine-tuned model using the test dataset.
    6. return the $v_f$ of every network traffic image in the test dataset.
```

IV. EVALUATION AND DISCUSSION

This section evaluates the performance of the proposed HAST-IDS by performing various experiments on DARPA1998 and ISCX2012, two commonly used public standard intrusion detection datasets. More specifically, these experiments aim to achieve the following:

- Evaluate the effect of network flow size on the performance of the HAST-IDS.
- Evaluate the effect of network packet size on the performance of the HAST-IDS.
- Evaluate the effect of the number of network packets of a network flow on the performance of the HAST-IDS.
- Evaluate the spatial-temporal features learned by the HAST-IDS using the t-SNE visualization approach.
- Show the best experimental results of the HAST-IDS and compare them with other published IDS techniques.

A. EXPERIMENTAL METHODOLOGY

1) DATASETS

The HAST-IDS learns features based on raw network traffic data; thus, the source dataset must contain raw traffic data. At present, most research methods use manually designed network traffic features. Thus, most public intrusion detection datasets, such as NSL-KDD [21] and Kyoto2009 [22], do not contain raw traffic data. From among the few public datasets that do contain raw traffic data, we choose DARPA1998 [23] and ISCX2012 [24] as our experiment datasets. The years of publication and the malware traffic types these datasets contain differ greatly; thus, they can be used to effectively evaluate the generality of the proposed method.

a: DARPA1998

In 1998, MIT's Lincoln Laboratory conducted an intrusion detection evaluation project funded by DARPA. One result of this project was a network traffic dataset simulating various intrusion behaviors, namely, DARPA1998. This dataset contains both normal traffic and four types of malware traffic (i.e., DoS, Probe, U2R, and R2L) and is divided into seven weeks of training traffic and two weeks of test traffic. The famous KDD99 dataset, which includes 41 manually designed traffic features, was derived from the DARPA1998 dataset and has become the most commonly used dataset in the field of intrusion detection and has used to produce numerous research results [25]. For comparison purposes, we choose DARPA1998 as one of the evaluation datasets.

The traffic format of DARPA1998 is non-split pcap, which must be split into multiple network flow files. In addition, the label files contain a few problems, such as duplicated records and incorrect labels. For example, the label file "Test/Week2/Friday" contains a record of "07/32/1998," which is an obvious date error. Therefore, the dataset requires preprocessing before the experiments can be conducted. First, the pkt2flow tool [26] is used to split the raw network traffic data into multiple network flows. Second, every label file is checked, and all duplicated records and incorrect records are removed. Finally, we match every network flow file to the processed label files. Table 1 shows the preprocessing results for the DARPA1998 dataset.

TABLE 1. Preprocessing results of the DARPA1998 dataset.

Dataset	Training		Test	
	Count	Percentage	Count	Percentage
Normal	849,991	34.46%	459,547	41.79%
DoS	1,561,231	63.29%	591,619	53.80%
Probe	48,984	1.99%	40,317	3.67%
R2L	6,494	0.26%	8,041	0.73%
U2R	229	0.01%	207	0.02%
Total	2,466,929		1,099,731	

b: ISCX2012

In 2012, the Information Security Center of Excellence (ISCX) of the University of New Brunswick (UNB) in Canada published an intrusion detection dataset named ISCX2012. This dataset contains seven days of raw network traffic data, including normal traffic and four types of attack traffic (i.e., BruteForce SSH, DDoS, HttpDoS, and Infiltrating). Some researchers have noted that the attack types considered in KDD99 are now obsolete [27]. In contrast, the attack types of ISCX2012 are more modern and closer to reality. In addition, the percentage of attack traffic is approximately 2.8%, which makes ISCX2012 similar to real-world datasets [28].

The ISCX2012 dataset also needs to be preprocessed before the experiments can be conducted. The preprocessing method is very similar to that used for DARPA1998. Because the traffic data of 16 June has only 11 attack flows, according to the provider's description, we removed these 11 flows and consider all traffic data of 16 June as normal. ISCX2012 was not divided by the provider into training and test datasets; therefore, we divide it into training and test datasets using a ratio of 60% to 40%, respectively. Moreover, this ratio has recently been used by many researchers, thus simplifying the comparison of our method with other methods. Table 2 shows the preprocessing results for the ISCX2012 dataset.

TABLE 2. Preprocessing results of the ISCX2012 dataset.

Dataset	Training		Test	
	Count	Percentage	Count	Percentage
Normal	890,726	97.27%	593,811	97.27%
BFSSH	4,179	0.46%	2,785	0.46%
Infiltrating	6,027	0.66%	4,017	0.66%
HttpDoS	2,090	0.23%	1,392	0.23%
DDoS	12,673	1.38%	8,448	1.38%
Total	915,695		610,453	

2) METRICS

Three metrics are used to evaluate the performance of the HAST-IDS: accuracy, detection rate (DR) and FAR, which are all commonly used in the field of intrusion detection. Accuracy is used to evaluate the overall performance of the system. DR is used to evaluate the system's performance with respect to its malware traffic detection. FAR is used to evaluate misclassifications of normal traffic. Their definitions are presented below. TP is the number of instances correctly classified as X, TN is the number of instances correctly classified as Not-X, FP is the number of instances incorrectly classified as X, and FN is the number of instances incorrectly classified as Not-X.

$$\begin{aligned} Accuracy(ACC) &= \frac{TP+TN}{TP+FP+FN+TN} \\ DetectionRate(DR) &= \frac{TP}{TP+FN} \\ FalseAlarmRate(FAR) &= \frac{FP}{FP+TN} \end{aligned} \tag{10}$$

In addition, an important goal of the HAST-IDS is to reduce the FAR as much as possible while improving the DR. To comprehensively evaluate the HAST-IDS considering both the DR and FAR, the effectiveness measure (EM) proposed by [29] is used in our research to compare the HAST-IDS with other published methods. The EM is slightly modified, and its definition is given below, where C is the number of classes and 1 denotes the normal class, which is excluded from the calculation. This formula results in a high value only when the DR is high and the FAR is low. In addition, when the test dataset size is bigger, the generalization capability of the system is better; thus, the EM value is larger. This metric, which considers the DR, FAR and test dataset size, can better reflect the comprehensive performance of the HAST-IDS. In addition, EM$_{DF}$ is also used, which is calculated using only the DR and FAR.

$$Effect.Mea.(EM) = \frac{\sum_{i=2}^{c} \frac{DR}{FAR}}{C-1} \times ln(\#Testing\ samples)$$

$$Effec.Mea._{DF}(EM_{DF}) = \frac{\sum_{i=2}^{c} \frac{DR}{FAR}}{C-1} \tag{11}$$

3) EXPERIMENTAL SETUP

Keras [30] and TensorFlow [31], which are run on the Ubuntu 16.04 64-bit OS, are used as the software frameworks. The server is a DELL R720 with 16 CPU cores and 16GB of memory. An Nvidia Tesla K40m GPU is used as the accelerator. Tables 3 and 4 describe the architectures of the deep neural networks of the HAST-I and HAST-II, respectively.

TABLE 3. DNN architectural parameters of HAST-I.

Layer	Type	Filters/neurons	Stride	Pad
1	conv+ReLU	32	1	same
2	max pooling	3	3	same
3	conv+ReLU	64	1	same
4	max pooling	3	3	same
5	dense	1,024	--	none
6	dense	5	--	none
7	softmax	--	--	none

B. INFLUENCE OF NETWORK FLOW SIZE

In HAST-I, each network flow must be reduced to a fixed size in the preprocessing stage due to the special requirements of CNN input data. Computer vision tasks do not have this problem because the size of every image or every video frame is fixed, and the images and video frames are easy to preprocess to an identical size. However, in the field of network traffic analysis, the number of packets in a flow is variable. Furthermore, the packet size is also variable, which causes large fluctuations among flow sizes. Table 5 shows the network flow size statistics of the two datasets.

As shown in Table 5, large differences in network flow sizes exist in both datasets. Thus, a method to choose a suitable flow size must be determined via the experiments. In our experiments, the flow size ranges from 100 to 1,500 bytes. At sizes above 1,500, the performance no longer improves. The evaluation metrics are accuracy, overall DR and overall FAR. Table 6 shows the experimental results. For the DARPA1998 dataset, the system achieves similar superior performances when the flow sizes are 800 and 1,000~1,300. Considering the training time, we choose 800 bytes as the best flow size. For the ISCX2012 dataset, the system achieves the best performance when the flow size is 600 bytes. When the flow size is larger than 700, the performance no longer improves. Tables 7 and 8 respectively show the best experimental results of HAST-I and HAST-II on the two datasets. Note that the two best sizes differ greatly from the mean sizes (57,719, 5,582) of the flows presented in Table 5. A possible explanation is that the information in the front part of a flow, which mainly corresponds to connection creation, may be more important when detecting malware traffic.

TABLE 4. DNN architectural parameters of HAST-II.

Layer	Type	Filters/neurons	Stride	Pad
1	conv+tanh	128	1	valid
2	max pooling	2	2	valid
3	conv+tanh	256	1	valid
4	max pooling	2	2	valid
5	dense	128	--	none
6	conv+tanh	192	1	valid
7	max pooling	2	2	valid
8	conv+tanh	320	1	valid
9	max pooling	2	2	valid
10	dense	128	---	none
11	lstm	92	---	none
12	lstm	92	---	none
13	dense	5	---	none
14	softmax	---	---	none

TABLE 5. Network flow size statistics.

Flow Length	DARPA1998	ISCX2012
Count	3,561,018	1,526,148
Mean	5,582	57,719
Max	116,690,260	1,276,599,890
Min	59	60
Median	120	3,019
Mode	60/21,268,245	186/10,199
Std	536	1,896,410

TABLE 6. Performance for different network flow sizes (%).

Flow	DARPA1998			ISCX2012		
Size	ACC	DR	FAR	ACC	DR	FAR
100	96.97	94.84	0.05	99.61	96.67	0.29
200	98.30	97.13	0.06	99.51	90.85	0.23
300	98.41	97.30	0.04	99.58	95.66	0.30
400	98.51	97.51	0.09	99.63	96.68	0.28
500	98.50	97.49	0.08	99.66	96.75	0.25
600	98.62	97.68	0.05	99.69	96.91	0.22
700	98.61	97.67	0.06	97.27	0.00	0.00
800	98.68	97.78	0.07	97.27	0.00	0.00
900	41.78	0.00	0.00	97.27	0.00	0.00
1,000	98.68	97.81	0.09	97.27	0.00	0.00
1,100	98.69	97.82	0.09	97.27	0.00	0.00
1,200	98.69	97.83	0.10	97.27	0.00	0.00
1,300	98.70	96.89	0.08	97.27	0.00	0.00
1,400	41.78	0.00	0.00	97.27	0.00	0.00
1,500	41.78	0.00	0.00	97.27	0.00	0.00

TABLE 7. Best performance of HAST-I for DARPA1998.

Dataset	ACC	DR	FAR
DoS	99.51	99.10	0.02
Probe	99.32	83.35	0.01
R2L	99.81	74.19	0.02
U2R	99.99	64.25	0.02
Overall	99.68	97.78	0.07

TABLE 8. Best performance of HAST-I for ISCX2012.

Dataset	ACC	DR	FAR
BFSSH	99.99	99.61	0.02
Infiltrating	99.98	97.68	0.04
HttpDoS	99.96	83.12	0.09
DDoS	99.76	97.94	0.07
Overall	99.69	96.91	0.22

The above analysis shows that even though many differences exist between the two datasets, both schemes achieve their highest accuracy, highest DR and lowest FAR when the flow size ranges from 600 to 800. An interesting finding is that applying a similar deep neural network architecture to classify the MNIST hand-written digit dataset also achieves very high accuracy [32]. The size of an MNIST image is 784 (28*28), which is very close to our chosen size. This similarity may indicate that the architecture of deep neural networks is suitable for detecting key information in data of that size, regardless of whether the data represents images or traffic.

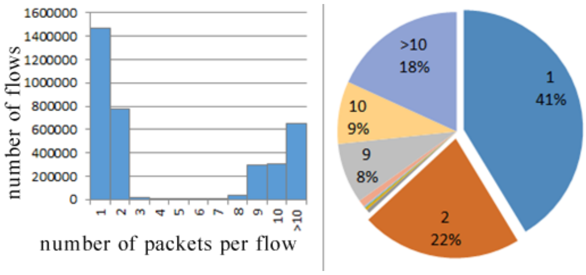


FIGURE 4. The number of packets per flow in the DARPA1998 dataset.

C. INFLUENCE OF NETWORK PACKET SIZE

The next two sections measure the effects of two key parameters of HAST-II on the system performance, namely, the packet size and the number of packets per flow. Note that the number of packets per flow is generally very small in the DARPA1998 dataset. Figure 4 shows that the majority of flows, i.e., 1,471,468, contain only one packet, accounting for 41.32% of the total, and 775,649 flows contain only two packets, accounting for 21.78% of the total. Together, these two flow sizes account for 63.1% of the total. The most important advantage of LSTM is its ability to learn the temporal features of a long sequence of data. However, the sequence length of the flows in the DARPA1998 dataset is too short. In particular, when the number of packets is 1, no sequence exists, which makes it meaningless to apply LSTM to the DARPA1998 dataset. Thus, the evaluation experiments for HAST-II are conducted only for ISCX2012.

During spatial feature learning in HAST-II, the input data unit of the CNNs is network packets. Similar to the discussion in section B, every packet must be reduced to a fixed size. In a network flow, the packet sizes generally differ greatly. Table 9 shows the statistics regarding packet sizes for the ISCX2012 dataset. As shown in Table 9, large differences in packet sizes exist in the ISCX2012 dataset. Section B discussed how flow size has an important effect on the system performance. Similarly, this section measures the effect of packet size on the performance of HAST-II and determines the best packet size.

TABLE 9. Statistics regarding network packet size for ISCX2012.

Packet Bytes	Value
Count	118,419,538
Mean	743
Max	1,514
Min	60
Median	598
Mode	60/39,805,790
Std	676

TABLE 10. Statistics regarding number of network packets per flow for ISCX2012.

Number of Packets per Flow	Value
Count	1,526,148
Mean	77
Max	1,306,267
Min	1
Median	14
Mode	10/177,277
Std	2,105

In our experiments, the packet sizes range from 100 to 1,000. We choose the median (14; see the median row in Table 10) as the number of packets per flow. The accuracy, DR and FAR of every class of traffic are used as evaluation metrics. Table 11 shows the experimental results, from which, we can see that those metrics yield the best results when the packet sizes are 100 and 200. Considering the training time, we choose 100 bytes as the best packet size, which differs greatly from the mean packet size of 743 shown in Table 9. A possible explanation is that the first 100 bytes mainly comprise the packet header, whose information may be more important for detecting intrusion.

An interesting finding is the comparison with the sentiment analysis task in the field of natural language processing. Using a similar deep neural network architecture, [15] chooses 512 characters as the best sentence size, which is equivalent to the packet size in our research. However, 512 characters cover the length of most sentences, whereas

TABLE 11. Effect of packet size on the performance of the HAST-IDS (%).

Packet	BFSSH			Infiltrating			HttpDoS			DDoS			Overall		
Size	ACC	DR	FAR	ACC	DR		ACC	DR	FAR	ACC	DR		ACC	DR	FAR
100	99.96	95.76	0.01	99.95	95.29	100	99.96	95.76	0.01	99.95	95.29	100	99.96	95.76	0.01
200	99.95	95.90	0.01	99.95	95.76	200	99.95	95.90	0.01	99.95	95.76	200	99.95	95.90	0.01
300	99.96	96.19	0.01	98.51	95.07	300	99.96	96.19	0.01	98.51	95.07	300	99.96	96.19	0.01
400	99.94	95.97	0.03	96.44	94.82	400	99.94	95.97	0.03	96.44	94.82	400	99.94	95.97	0.03
500	99.93	98.99	0.01	99.90	86.08	500	99.93	98.99	0.01	99.90	86.08	500	99.93	98.99	0.01
600	99.92	98.59	0.01	99.90	85.98	600	99.92	98.59	0.01	99.90	85.98	600	99.92	98.59	0.01
700	99.96	96.55	0.01	99.91	96.31	700	99.96	96.55	0.01	99.91	96.31	700	99.96	96.55	0.01
800	99.96	96.05	0.01	99.94	93.45	800	99.96	96.05	0.01	99.94	93.45	800	99.96	96.05	0.01
900	99.96	96.05	0.01	99.94	93.45	900	99.96	96.05	0.01	99.94	93.45	900	99.96	96.05	0.01
1,000	99.94	98.92	0.01	99.75	88.00	1,000	99.94	98.92	0.01	99.75	88.00	1,000	99.94	98.92	0.01

TABLE 12. Effect of number of packets on the performance of the HAST-IDS (%).

Number of Packets	BFSSH			Infiltrating			HttpDoS			DDoS			Overall		
	ACC	DR	FAR	ACC	DR		ACC	DR	FAR	ACC	DR		ACC	DR	FAR
6	99.96	97.09	0.01	99.96	96.21	6	99.96	97.09	0.01	99.96	96.21	6	99.96	97.09	0.01
8	99.96	96.48	0.01	99.95	94.97	8	99.96	96.48	0.01	99.95	94.97	8	99.96	96.48	0.01
10	99.97	96.98	0.01	99.95	95.61	10	99.97	96.98	0.01	99.95	95.61	10	99.97	96.98	0.01
12	99.96	96.55	0.01	99.95	95.22	12	99.96	96.55	0.01	99.95	95.22	12	99.96	96.55	0.01
14	99.96	96.37	0.01	99.95	95.24	14	99.96	96.37	0.01	99.95	95.24	14	99.96	96.37	0.01
16	99.96	96.19	0.01	99.94	93.67	16	99.96	96.19	0.01	99.94	93.67	16	99.96	96.19	0.01
18	99.96	95.61	0.01	99.95	95.54	18	99.96	95.61	0.01	99.95	95.54	18	99.96	95.61	0.01
20	99.94	98.74	0.01	99.87	82.07	20	99.94	98.74	0.01	99.87	82.07	20	99.94	98.74	0.01
22	99.95	96.01	0.02	99.68	90.88	22	99.95	96.01	0.02	99.68	90.88	22	99.95	96.01	0.02
24	99.96	95.90	0.01	99.93	91.63	24	99.96	95.90	0.01	99.93	91.63	24	99.96	95.90	0.01
26	99.95	96.80	0.02	99.94	95.41	26	99.95	96.80	0.02	99.94	95.41	26	99.95	96.80	0.02
28	99.97	98.16	0.01	99.95	94.24	28	99.97	98.16	0.01	99.95	94.24	28	99.97	98.16	0.01
30	99.95	96.91	0.01	99.91	94.00	30	99.95	96.91	0.01	99.91	94.00	30	99.95	96.91	0.01

the 100 bytes used in our study covers only the first small part of a network packet. Intuitively, this result may occur because the structure of a sentence is very different from that of a packet. The information in a sentence tends to follow a uniform distribution, whereas a packet can be divided into a packet header and payload, thus, its information tends to be emphasized differently.

D. INFLUENCE OF THE NUMBER OF NETWORK PACKETS

In the temporal feature learning stage of HAST-II, LSTM requires that the number of packets per flow be fixed. However, the number of packets per flow generally differs greatly. Table 10 shows the statistics regarding the number of packets per flow for the ISCX2012 dataset. The table shows that in practice, the number of packets per flow varies widely. Therefore, we must measure the effect of the number of packets per flow on the system performance and determine the best number of packets via multiple evaluation experiments.

In our experiments, the number of packets ranges from 6 to 30. When the number of packets is larger than 30, the performance no longer improves. The packet size is 100 bytes, as determined in section C. The accuracy, DR and FAR of every class of traffic are used as evaluation metrics. Table 12 shows the experimental results. From the table, we see that those metrics yield the best results when the number of packets is 6, 8 and 14. Considering the training time, we choose 6 as the best number of packets. This number differs greatly from the mean number of packets (77) shown in Table 10. This result probably occurs because the first few packets in a flow correspond with connection creation, whose information may be more important for detecting intrusion behaviors. This fact and its possible explanation are both very similar to those of section B. Celik *et al.* [33] reported similar findings (i.e., the first few packets play a relatively more important role in malware traffic detection).

Via the experiments presented in sections C and D, we finally obtain the best values for the two parameters of HAST-II for the ISCX2012 dataset, namely, the best packet size (100) and the best packet number (6). The experimental results obtained using those two parameters are shown in Table 13. Compared with Table 8, Table 13 shows that the accuracy and DR are approximately equal, while the FAR is remarkably reduced. We conclude that HAST-II achieves a comprehensively better performance than that of HAST-I on the ISCX2012 dataset, which indicates that spatial-temporal features are more effective than single spatial features in reducing the FAR on the ISCX2012 dataset.

TABLE 13. Best performance of HAST-II for ISCX2012.

Dataset	ACC	DR	FAR
BFSSH	99.96	97.09	0.01
Infiltrating	99.96	96.21	0.00
HttpDoS	99.96	92.88	0.00
DDoS	99.95	97.95	0.00
Overall	99.89	96.96	0.02

E. VISUALIZATION OF SPATIAL-TEMPORAL FEATURES USING T-SNE

The t-SNE algorithm [34] is used in this section to perform dimensionality reduction and visualization analysis for the spatial-temporal traffic features learned by the HAST-IDS. The t-SNE algorithm is a nonlinear dimensionality reduction algorithm. Compared to linear dimensionality reduction algorithms such as principal component analysis (PCA), t-SNE can obtain better low-dimensional results and better visualizations. According to the results discussed in sections B, C and D, we apply the t-SNE algorithm to the high-dimensional result vectors of the two best experiments, namely, the network flow vectors learned by HAST-I for DARPA1998 and by HAST-II for ISCX2012.

Specifically, the flow vectors learned by the CNN in HAST-I or by the LSTM in HAST-II are saved before application of the softmax classifier. The flow vector is a five-dimensional vector. In each test dataset of the two datasets, 500 samples are randomly selected from every class of traffic. The total number of samples in the ISCX2012 dataset is 2,500; however, because there are only 207 total test samples of U2R traffic, the total number of samples from the DARPA1998 dataset is 2,207. The t-SNE source code file we used was sourced from [35]. Because the flow vectors are only five-dimensional, we do not use the PCA preprocessing stage of that code file; instead, we directly apply t-SNE to reduce the dimensionality. The visualization results for the resulting low-dimensional vectors are shown in Figures 5 and 6.

From Figure 5, for the DARPA1998 dataset, the cluster effects for three classes of traffic, i.e., Normal, DoS, and Probe, are very good. The experimental data also yield the same result. The cluster effects for the R2L and U2R classes of traffic are not as good as those of the other three classes. Although a cluster effect exists, there are too many clusters, and their distances from other clusters are too small to clearly distinguish between clusters. The experimental data also yield the same results. This outcome possibly occurs because there are too few training samples of R2L and U2R compared to those of the other three classes. The numbers of training samples of the Normal, DoS, and Probe classes are 849,991, 1,561,231 and 48,984, respectively, whereas the R2L and U2R classes have only 6,494 and 229 training samples, respectively. The imbalance of training data [36] results in the HAST-IDS being unable to learn sufficient representative features, which causes a poor detection performance for these two classes. Figure 6 shows a good overall clustering effect for the ISCX2012 dataset, although the discrimination degrees of some classes, such as DDoS and Infiltrating, are not very high. The reason is also most likely due to an imbalance in the training samples. In general, the visualization results of the two datasets using the t-SNE algorithm are quite good, which directly demonstrates the capability of the HAST-IDS to learn spatial-temporal features and indirectly explains why the subsequent softmax classifiers achieve such a high performance.

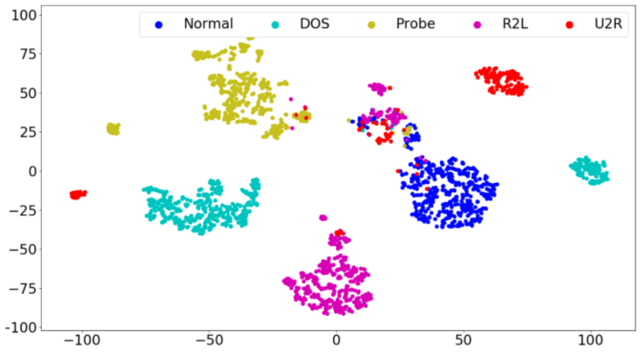


FIGURE 5. Visualization result of flow vectors for DARPA1998.

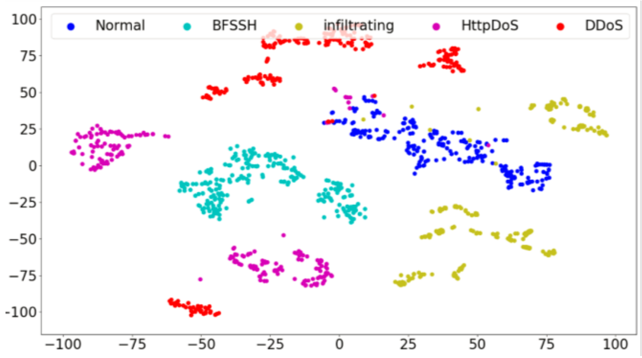


FIGURE 6. Visualization result of flow vectors for ISCX2012.

F. COMPARISON WITH OTHER PUBLISHED METHODS

Researchers have proposed many intrusion detection methods, most of which are based on manually designed traffic features. In contrast, the method proposed in this paper learns spatial-temporal traffic features directly from raw network traffic data. We expect the automatically learned traffic features be more accurate and more representative, thus, we compare the experimental results of the HAST-IDS with those of other published methods.

Table 14 shows the experimental results comparison for the DARPA1998 dataset. The compared methods are all based on KDD99, which is a dataset containing 41 manually designed

TABLE 14. Comparison with other published methods for DARPA1998.

Method	DoS (%)		Probe (%)		R2L (%)		U2R (%)		Test Dataset Size	EM_{DF}	EM
	DR	FAR	DR	FAR	DR	FAR	DR	FAR			
PLSSVM[37]	78.69	0.73	86.46	13.87	84.85	0.53	30.7	0.47	311,029	8.49E+01	1.07E+03
Multi-Classifier[38]	97.3	0.4	88.7	0.4	9.6	0.4	29.8	0.1	311,029	1.97E+02	2.49E+03
Random Forest[39]	98.91	3.15	55.12	0.45	66.67	5.18	100	0.13	77,287	2.34E+02	2.63E+03
Bayes Net[40]	94.6	0.2	83.8	0.13	5.2	0.6	30.3	0.3	15,437	3.07E+02	2.96E+03
JRip[40]	97.4	0.3	83.8	0.1	0.1	0.4	12.8	0.1	15,437	3.23E+02	3.11E+03
SVM[41]	76.7	0.09	81.2	0.36	11.2	0.08	21.4	0.08	10,000	3.71E+02	3.42E+03
Naive Bayes[42]	99.69	0.04	99.11	0.45	99.11	8.02	64	0.14	311,029	7.95E+02	1.01E+04
ID3[43]	99.9	0.03	99.7	0.55	93.5	0.98	49.1	0.15	311,029	9.84E+02	1.24E+04
EID3[43]	99.9	0.03	99.8	0.39	99.7	0.22	99.8	0.12	311,029	1.22E+03	1.54E+04
MARK-ELM[29]	99.96	0.02	97.42	0.03	94.94	0.05	62.87	0.01	72,793	4.11E+03	4.60E+04
HAST-IDS	99.10	0.02	83.35	0.01	74.19	0.02	64.25	0.02	1,099,731	5.05E+03	7.03E+04

TABLE 15. Comparison with other published methods for ISCX2012 (%).

Method	Normal-DR	Attack-DR	Accuracy	FAR
MHCVF[44]	99.9	68.2	99.5	0.03
ALL-AGL[45]	99.5	93.2	95.4	0.30
KMC+NBC[46]	97.7	99.7	99.0	2.2
AMGA2-NB[11]	95.2	92.7	94.5	7.0
HAST-IDS	99.97	96.96	99.89	0.02

TABLE 16. Comparison of training time and testing time achieved using HAST-IDS with those of other published methods for DARPA1998.

Method	Training time	Testing time
MHCVF[44]	240 min	35 min
PLSSVM[37]	104 min	46 min
Multiple-Level Hybrid Classifier (MLHC)[47]	1,443 min	4 min
HAST-IDS	58 min	1.7 min

features extracted from the DARPA1998 dataset. Accuracy and EM, proposed in section A, are used as the evaluation metrics. The EM metric comprehensively considers three metrics, DR, FAR and test dataset size, and can better measure the system performance. For example, the DR of the Probe class of the EID3 method is very high, but the FAR is too high, while the FAR of the U2R class of the SVM method is very low, but the DR is too low. The EM metric can be used to comprehensively compare these methods more fairly. Table 14 shows that the HAST-IDS does not always perform best on every evaluation metric. For example, the DR of the DoS class is lower than that of the EID3 method, and the FAR of U2R is higher than that of MARK-ELM. However, the HAST-IDS achieves the best EM result among all listed methods. Even without considering the test dataset size, the EM_{DF} result of the HAST-IDS is still the best. Thus, our method achieves the best comprehensive performance among all methods. It learns more accurate features, and better features effectively result in a reduced FAR, which is consistent with the conclusion of [8].

Table 15 shows a comparison of the experimental results for the ISCX2012 dataset. This dataset was published much later than DARPA1998; thus, there are relatively fewer available corresponding experimental results. The methods listed in Table 15 all use 16 manually designed traffic features. Because most of these methods do not report the accuracy and FAR for every traffic class, we cannot compare their EM values. Based on the available experimental results for the compared methods, the DR of normal traffic, DR of attack traffic, accuracy and overall FAR are used as the evaluation metrics. Table 15 shows that the HAST-IDS achieves the best performances regarding the DR of normal traffic, DR of attack traffic, and overall FAR, exceeding those of the other state-of-the-art methods by 0.07%, 0.39% and 0.01%, respectively. The DR of attack traffic obtained by the HAST-IDS is lower than that of the KMC+NBC method by 2.74% and ranks second among all five methods.

Regarding the training and testing times, the input data for the HAST-IDS consists of raw network traffic; thus, the training and testing times of our method include the time required for feature extraction and feature selection. In contrast, the previously mentioned methods directly use manually designed features and do not require time for feature extraction and selection. Thus, it is not suitable to compare their training and testing times directly. However, we do list the training and testing times that exist in the literature for some methods. The experimental hardware we used is listed in Section A. Table 16 shows the comparison of the training and testing times for the DARPA1998 dataset. The authors of the MHCVF and MLHC methods did not report the FAR for every traffic class, and the corresponding EM values cannot be calculated; thus, they do not appear in Table 14. Because the ISCX2012 dataset is relatively very new and insufficient studies exist that reports training and testing times, we cannot perform a comparison study. Table 16 shows that the training and testing times of the HAST-IDS for the entire DARPA1998 dataset are 58 min (3,533 s) and 1.7 min (103 s), respectively. The table also shows that although the HAST-IDS times include feature extraction and selection, it still achieves the lowest training and testing times, 46 min and 2.3 min, respectively, compared to those of the previously mentioned state-of-the-art methods, which clearly shows the high efficiency of our proposed method for intrusion detection.

V. CONCLUSION AND FUTURE WORK

Because of the difficulty of hand-designing accurate traffic features in the field of intrusion detection, we propose the HAST-IDS, which uses deep neural networks that can automatically learn hierarchical spatial-temporal features directly from raw network traffic data. To the best of our knowledge, this is the first time that a representation/feature-learning method based on raw traffic data has been applied in the field of intrusion detection. The method uses CNNs to learn the spatial features of network packets and then uses an LSTM to learn the temporal features among multiple network packets. As a result, the proposed method obtains more accurate spatial-temporal traffic features. The method does not require any of the engineering techniques used in traditional intrusion detection methods. The experimental results show that the HAST-IDS effectively improves the accuracy and DR compared to other published methods. In addition, the FAR of many current intrusion detection methods is generally high. Eesa *et al.* [8] showed that the detection rate can be increased while the FAR can be decreased by using a better traffic feature set. Our experimental results show that the HAST-IDS effectively reduces the FAR because it automatically learns the spatial-temporal features, which improve the overall performance of the IDS.

Two problems require further study in future work. The first involves improving the detection performance on imbalanced datasets [36]. In the real world, the amount of malware traffic is small compared to the amount of normal traffic, and the proportions of different classes of malware traffic often differ greatly. The t-SNE visualization results and experimental data both show that the performance of the HAST-IDS is not good enough for the classes of traffic with fewer samples. We will focus on that problem in future work. The second problem involves combining traditional traffic features. Many published research results show that in certain cases, some manually designed traffic features can be very useful. To further improve the system performance, the usefulness of either an integration of those features into the HAST-IDS framework or the use of an ensemble learning method is worth exploring.

Combining the previous research results [13], [48], we conclude that deep neural networks can automatically learn features directly from raw network traffic data and achieve good results in the field of intrusion detection or network anomaly detection. The preliminary experimental results are promising. Following up on this idea, we will continue to research the application of deep neural networks in the IDS field with the goal of further improving IDS performance.

REFERENCES

[1] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[2] F. Zhang and D. Wang, "An effective feature selection approach for network intrusion detection," in *Proc. IEEE 8th Int. Conf. Netw., Archit. Storage*, Xi'an, China, Jul. 2013, pp. 307–311.

[3] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Comput. Commun.*, vol. 49, pp. 1–17, Aug. 2014.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[5] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[6] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, "A deep learning approach for network intrusion detection system," *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BIONETICS)*, 2015, pp. 21–26.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[8] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2670–2679, Apr. 2015.

[9] J. Gu *et al.*. (2017). "Recent advances in convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1512.07108

[10] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Intell. Res.*, vol. 57, pp. 345–420, Nov. 2016.

[11] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2519–2533, Sep. 2015.

[12] D. Ariu, R. Tronci, and G. Giacinto, "*HMMPayl*: An intrusion detection system based on hidden Markov models," *Comput. Secur.*, vol. 30, no. 4, pp. 221–241, Jun. 2011.

[13] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. 31st Int. Conf. Inf. Netw. (ICOIN)*, Da Nang, Vietnam, Jan. 2017, pp. 712–717.

[14] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for Botnet detection behavior," in *Proc. IEEE Biennial Congr. Argentina (ARGENCON)*, Buenos Aires, Argentina, Jun. 2016, pp. 1–6.

[15] S. Vafeias. *Character Level Models for Sentiment Analysis*. Accessed: Sep. 1, 2017. [Online]. Available: https://github.com/offbit/char-models

[16] J. Li, H. Xu, X. He, J. Deng, and X. Sun, "Tweet modeling with LSTM recurrent neural networks for hashtag recommendation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, Jul. 2016, pp. 1570–1577.

[17] X. Zhang and Y. LeCun. (Apr. 2016). "Text understanding from scratch." [Online]. Available: http://arxiv.org/pdf/1502.01710v5

[18] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 569–575, Mar. 2010.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 807–814.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6.

[22] J. Song, H. Takakura, and Y. Okabe. *Description of Kyoto University Benchmark Data*. [Online]. Available: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf

[23] R. P. Lippman *et al.* (1998). *Results of the DARPA 1998 Offline Intrusion Detection Evaluation*. [Online]. Available: https://ll.mit.edu/ideval/files/RAID_1999a.pdf

[24] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.

[25] S. Devaraju and S. Ramakrishnan, "Performance comparison for intrusion detection system using neural network with KDD dataset," *ICTACT J. Soft Comput.*, vol. 4, no. 3, pp. 743–752, 2014.

[26] X. Chen. *A Simple Utility to Classify Packets Into Flows*. Accessed: Sep. 1, 2017. [Online]. Available: https://github.com/caesar0301/pkt2flow

[27] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[28] J. Song, H. Takakura, Y. Okabe, and Y. Kwon, *Correlation Analysis Between Honeypot Data and IDS Alerts Using One-class SVM*. Rijeka, Croatia: InTech, 2011.

[29] J. M. Fossaceca, T. A. Mazzuchi, and S. Sarkani, "MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection," *Expert Syst. Appl.*, vol. 42, no. 8, pp. 4062–4080, May 2015.

[30] F. Chollet. *Keras*. Accessed: Sep. 1, 2017. [Online]. Available: https://github.com/fchollet/keras

[31] M. Abadi *et al.* (Mar. 2016). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1603.04467

[32] Y. A. LeCun *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Netw.*, vol. 8, no. 1, pp. 261–276, 1995.

[33] Z. B. Celik, R. J. Walls, P. McDaniel, and A. Swami, "Malware traffic detection using tamper resistant features," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Tampa, FL, USA, Oct. 2015, pp. 330–335.

[34] L. van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[35] L. van der Maaten. *Python Implementation of t-SNE*. Accessed: Sep. 1, 2017. [Online]. Available: https://lvdmaaten.github.io/tsne

[36] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[37] F. Amiri, M. M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1184–1199, Jul. 2011.

[38] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proc. Int. Conf. Mach. Learn., Models Technol. Appl.*, Las Vegas, NV, USA, 2003, pp. 209–215.

[39] M. Al Mehedi Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support vector machine and random forest modeling for intrusion detection system (IDS)," *J. Intell. Learn. Syst. Appl.*, vol. 6, no. 1, pp. 45–52, 2014.

[40] H. A. Nguyen and D. Choi, "Application of data mining to network intrusion detection: Classifier selection model," in *Challenges for Next Generation Network Operations and Service Management* (Lecture Notes in Computer Science), vol. 5297. Berlin, Germany: Springer-Verlag, 2008, pp. 399–408.

[41] X. Xu, "Adaptive intrusion detection based on machine learning: Feature extraction, classifier construction and sequential pattern prediction," *Int. J. Web Services Practices*, vol. 2, nos. 1–2, pp. 49–58, 2006.

[42] D. M. Farid, N. Harbi, and M. Z. Rahman, "Combining Naïve Bayes and decision tree for adaptive intrusion detection," *Int. J. Netw. Secur. Appl.*, vol. 2, no. 2, pp. 189–196, 2010.

[43] V. Jaiganesh, P. Mangayarkarasi, and P. Sumathi, "An efficient algorithm for network intrusion detection system," *Int. J. Comput. Appl.*, vol. 90, no. 12, pp. 12–16, Mar. 2014.

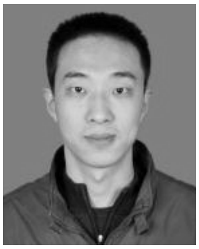
[44] A. Akyol, M. Hacibeyoğlu, and B. Karlik, "Design of multilevel hybrid classifier with variant feature sets for intrusion detection system," *IEICE Trans. Inf. Syst.*, vol. E99-D, no. 7, pp. 1810–1821, 2016.

[45] H. Sallay, A. Ammar, M. Ben Saad, and S. Bourouis, "A real time adaptive intrusion detection alert classifier for high speed networks," in *Proc. IEEE 12th Int. Symp. Netw. Comput. Appl.*, Cambridge, MA, USA, Aug. 2013, pp. 73–80.

[46] W. Yassin, N. I. Udzir, Z. Muda, and M. N. Sulaiman, "Anomaly-based intrusion detection through K-means clustering and Naives Bayes classification," in *Proc. 4th Int. Conf. Comput. Inform. (ICOCI)*, 2013, pp. 298–303.

[47] C. Xiang, P. C. Yong, and L. S. Meng, "Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees," *Pattern Recognit. Lett.*, vol. 29, no. 7, pp. 918–924, May 2008.

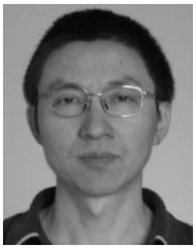
[48] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Inform. (ISI)*, Beijing, China, Jul. 2017, pp. 43–48.



WEI WANG received the B.S. and M.S. degrees from PLA Information Engineering University, Zhengzhou, China, in 2006 and 2011, respectively. He is currently pursuing the Ph.D. degree with the University of Science and Technology of China, Hefei, China. His research interests include machine learning and cyberspace security.

YIQIANG SHENG received the master's degree from Nankai University, Tianjin, China, in 2003, and the Ph.D. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 2014. He is currently with the National Network New Media Engineering Research Center, Chinese Academy of Sciences, Beijing, China, as an Academic Researcher and an Associate Professor. His current research interests include smart systems, optimization algorithms, machine learning, big data, and network theory with applications.

JINLIN WANG received the B.S. degree in mathematics from the University of Science and Technology of China in 1986, and the M.S. degree in acoustics from the Institute of Acoustics, Chinese Academy of Sciences (IACAS), in 1989. He is currently a Research Professor at the IACAS and the Director of the National Network New Media Engineering Research Center. His current research interests include the structure and new service of broadband networks, digital media services, and network architecture.



XUEWEN ZENG received the B.S. degree in automatic control from Shanghai Jiao Tong University, China, in 1989, and the M.S. and Ph.D. degrees in signal and information processing from the Institute of Acoustics, Chinese Academy of Sciences (IACAS), in 1992 and 1997, respectively. He is a Research Fellow with IACAS and a Professor at the Chinese Academy of Sciences. His research interests include multimedia communication and pattern recognition.

XIAOZHOU YE received the M.S. degree from the Beijing University of Posts and Telecommunications in 2004, and the Ph.D. degree from the Institute of Acoustics, Chinese Academy of Sciences (IACAS) in 2007. He is currently an Associate Professor at IACAS. His research activities involve application delivery networks, embedded network processors and intelligent terminal systems, home network media gateways, broadband access networks, and streaming media processing.

YONGZHONG HUANG received the Ph.D. degree from the National Digital Switching System Engineering and Technological Research and Development Center in 2007. He is currently a Professor at the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China. His research interests are high-performance computing, big data, and artificial intelligence.

MING ZHU received the B.S., M.S., and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 1986, 1989, and 2002, respectively. He is currently a Professor with the Department of Automation, University of Science and Technology of China. His research interests generally lie in the fields of data mining, pattern recognition, and machine learning.