IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2019, accepted June 18, 2019, date of publication June 24, 2019, date of current version July 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924492

A Packet-Length-Adjustable Attention Model Based on Bytes Embedding Using Flow-WGAN for Smart Cybersecurity

LUCHAO HAN, YIQIANG SHENG, AND XUEWEN ZENG

National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Yiqiang Sheng (shengyq@dsp.ac.cn)

This work was supported in part by the Chinese Academy of Sciences through the Strategic Priority Research Program under Grant XDC02010701, and in part by the National Science and Technology Major Project of the Ministry of Science and Technology, China.

ABSTRACT In the studies of cybersecurity, malicious traffic detection is attracting more and more attention for its capability of detecting attacks. Almost all of the intrusion detection methods based on deep learning have poor data processing capacity with the increase in the data length. Most intrusion detection methods can only handle the header part of the traffic and omit valuable information from the payload, so they cannot detect the malicious traffic when the hacker hides the attack behavior in the payload. In this paper, we propose an attention model that can process network traffic flow with adjustable length to detect the payload-based attacks. Furthermore, we design a Flow Wasserstein GAN model to generate new network traffic data from the original data sets to enhance the network packet data and protect the user's privacy. Our model has a hierarchical structure to build the representations of bytes and packets on two levels. Moreover, two levels of attention mechanisms enable the model to pay attention to more important content when constructing the flow representation. The experiments based on the ISCX-2012 and ISCX-2017 datasets prove that the proposed model has higher performance in accuracy and true positive rate (TPR) than four state-of-the-art deep learning methods. The experiment shows that our model outperforms the existing HSAT-IDS in the detection of the generated packets.

INDEX TERMS Malicious traffic detection, deep learning, cybersecurity.

I. INTRODUCTION

Malicious network traffic classification is one of the cybersecurity task [1]. The goal of it is to assign labels to flow or packets. Various machine learning algorithms such as support vector machine (SVM) [2], Bayesian network [3] and artificial neural network (ANN) [4] are used as classifiers of malicious network traffic data. The traffic data contains plenty of noise, and the input of specific features can significantly increase the accuracy of intrusion and malware detection. However, some features may decrease the detection accuracy [5]. Therefore, it is indispensable for intrusion detection to do feature engineering. The selection of features is a challenging, and the complexity of the feature selection is NP-Hard [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenyu Zhou.

Although many deep learning methods approach to malicious traffic detection have been quite effective [7]–[9], in this paper we evaluate the hypothesis that better representations can be obtained by incorporating information of network data structure in the model architecture. The hypothesis in our model is that not all parts of network traffic flow are equally relevant for malicious network classification. And determining the relevant sections in a flow involves modeling the interactions of the bytes.

In addition to that, since deep learning methods take the header of the traffic flow as input, with the length of the input increase, the performance of these methods will inevitably decline. For some types of attacks such as DoS attack, it is difficult to judge whether the flow is malicious by only using the first few packets in a flow. So it is necessary to increase the length of the input. Using the attention mechanism our model can process the input with long length well and pay

attention to the crucial part to determine whether this traffic is malicious traffic. We consider to use bidirectional Gated Recurrent Units(GRU) to build the relationships between non-adjacent parts which can be very important.

Recurrent Neural Network (RNN), Long short-term memory (LSTM) [10] and GRU [11] have been absolutely established as the most advanced method for sequence modeling and conversion problems such as natural language processing. The attention mechanism becomes very important because it in the natural language processing frees the network from having to map a sequence of random length to a single, fixed-dimensional vector. In this work, we propose a classification model based on the attention mechanism to describe global interactions between input and output. Using the attention mechanism our model can process a long flow and find the critical parts for the classification task.

Word embedding also known as distributed word representation, [12] can capture the semantic and syntactic information of words from a large unlabeled corpus [13] and has drawn great attention from many researchers. Many models which are proposed in recent years have yielded most advanced results in various natural language processing tasks. Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing [14] and sentiment analysis [15]. In this work, we use the word embedding method to do the distributed bytes representation. Bytes embedding is based on the hypothesis which is similar to word embedding: bytes that occur in similar contexts tend to have similar meanings.

The main contributions of this paper are as follows. (1) In network traffic, different binary bytes have different meanings, and they are also related to each other. However, the one-hot encoding(OHE) does not reflect the association between them but uses the word embedding method to convert the bytes into byte vectors. In this article, we process the network flow as natural language and word2vector [16] to embedding the bytes to bytes vector. So the distance between any two vectors (eg, L2 distance or cosine distance) can represent a partial semantic relationship of two associated bytes. (2) We propose a hierarchical attention model which can learn information from two levels of the network flow structure. Our model first builds representations of bytes using bidirectional GRU and assign different bytes to different weights through the attention mechanism [17]. So some critical bytes are assigned more weight compared to some other bytes which are irrelevant to classifying this flow. Then we process the packet vector as same as processing bytes. We use bidirectional GRU to aggregate the representations of bytes to a flow representation since it makes full use of the context information from both directions, and any packet is also assigned a weight to let the model pay more or less attention to a different packet when building the representation of the flow. In this article, we show that the hierarchical embedding attention network often results in better performance. (3) The problem of missing data is frequently encountered in intrusion detection studies. Especially in the deep learning method, the limitations of the training data severely limit the training effect of the model. Furthermore, detect real user network traffic directly may cause a user data breach. So we propose a Flow-WGAN(Wasserstein GAN [18]) to generate new data from the original data set. This method cannot create new information, but Flow-WGAN can learn different feature from the same original training set. The reason is that the two models have different structures and the methods for extracting information are different. So the generated network flow packets are new data for the classifier model. Therefore we can use the generated network flow packet to simulate a new type of Internet application to evaluate the performance of the classifiers or improve the classifiers. The experiment shows that all classifiers have a higher FAR when process the generated network flow packets and our model has a lower FAR than the HAST-IDS model.

II. RELATED WORK

A. INTRUSION DETECTION

With the development of the Internet, network security has become one of the most important factors in various industries, such like industrial automation [19], [20], smart grid [21]. In the studies of cybersecurity, malicious traffic detection attracts more and more attention. Intrusion and malicious detection can be classified into two types of detection: signature-based and anomaly-based detection [22]. Signature-based detection, which is also known as misuse detection, has the advantage of a high detection rate and low false discovery rate. The disadvantage of signature-based is that it cannot detect new types of attacks or encrypted malicious network data [23]. As for anomaly-based detection, which is also called the statistical-based method, it can detect unknown attacks and encrypted malicious network data when it combined the latest technology such as machine learning. If using traditional machine learning, extracting features is necessary so a model can be built based on these features and be trained supervised or unsupervised. However, designing a set of suitable feature is very difficult [24]. And anomaly-based detection highly depends on feature designing. There is no rule guiding people to design feature set, so security practitioner can only design feature set based on previous experience. That causes the low accuracy problem of this approach.

B. DEEP LEARNING

In recent years, deep artificial neural networks have won numerous contests in pattern recognition and machine learning [25], [26]. One of the most significant advantages of it is that it can directly learn from the raw data, so there is no need to extract feature manually. Deep learning has made great achievement in many fields such as computer vision and natural language processing [27]. In the field of malicious traffic detection, more and more researchers have used deep learning algorithms and got remarkable results.

Ma et al. deployed deep learning algorithms using KDD99 [28]. Further on this basis, Kolosnjaji *et al.* [29] verified that CNN could identify malware network flow by using raw network data. Wang *et al.* [7] were inspired by the language model and process the raw network as a language. Wang *et al.* [30] proved that CNN could treat the network as the image and achieve high precision in detecting malicious network flow. Shabtai *et al.* [31] present mobile malware detection through analysis of deviations in application network behavior used semi-supervised machine-learning method to detect meaningful deviations in a mobile application's network behavior.

In [5], a novel intrusion detection system with genetic-algorithm-based feature selection and multiple support vector machine classifiers for wireless mesh networks are proposed. Reference [8] evaluates methods for applying unsupervised anomaly detection to cybersecurity applications on computer network traffic data, or flow. In [32], researchers present a novel adaptive and functionally auto-scaling stream processor: ''Wisdom'' with a built-in hybrid optimizer developed using Particle Swarm Optimization, and Bisection algorithms to optimize CEP rule parameters. In [33], all publicly accessible and labeled NIDS data sets are explored. In [34], a distributed attack detection system is proposed, and it has great performance in the experiment. Reference [35] proposed a single-stage malware classifier based on a character-level convolutional neural network. Reference [36] proves that an entropy-based approach is suitable to detect modern botnet-like malware based on anomalous patterns in the network. Reference [37] compares OPFC with data clustering based on k-means, and self-organization maps in the context of intrusion detection in computer networks. In the [38] the proposed system treats traffic records as images and detection of DoS attacks as a computer vision problem. Their method can process 59,000 traffic records per second. In [39] they tokenize and compress network flow into sequences of ''words'' that form ''sentences'' representative of a conversation between computers. In this way, they demonstrate positive unsupervised attack identification performance on the ISCX IDS dataset.

C. GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) [40] have shown remarkable results in various tasks such as image generation [41]–[44], image translation [45]–[47], super-resolution imaging [48], and face image synthesis [49]–[52]. The generative adversarial network (GAN) contains two parts: a generator and a discriminator. The discriminator reveals whether a sample is fake or real, while the generator produces samples as real as possible to cheat the discriminator. This approach also leverages the loss function to make the generated samples as realistic as possible.

D. GATED RECURRENT UNIT

Gated Recurrent Unit is a slightly more dramatic variation on the LSTM, introduced by Cho *et al.* [53]. It combines the ''forget'' gate and ''input'' gate into a single ''update gate,'' and it also merges the hidden state and cell state. So GRU is simpler than standard LSTM models and is easy to be trained. The GRU has the reset gate r_t and the update gate z_t. They determine how to update the state and combine state with the new input.

In Fig. 1. the update gate z_t decides how much past memory is retained. It is updated as flow:

$$z_t = sigmoid(W_z \cdot [h_{t-1}, x_t]). \tag{1}$$

In equation1 h_t is the hidden state and x_t is input at time t. The reset gate r_t decides how much past memory is combined with the new input. It is updated as flow:

$$r_t = sigmoid(W_r \cdot [h_{t-1}, x_t]). \tag{2}$$

The candidate hidden state $\widetilde{h}_t$ is computed with the reset gate, just like traditional RNN method.

$$\widetilde{h}_t = tanh(W \cdot [r_t \times h_{t-1}, x_t]). \tag{3}$$

The new state ht is computed as:

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \widetilde{h}_t. \tag{4}$$

In the initial state, the reset gate is zero, so the past memory can be ignored.

E. ATTENTION MECHANISM IN NETWORK

Inspired by human attention mechanism [54], application of attention mechanism has achieved great success in natural language processing and computer vision task. Instead of compressing the entire image or a sequence into static representation, using attention mechanism network can focus on the most critical part of features as needed. Reference [55] presents a recurrent neural network model that is capable of extracting information from an image or video by adaptively selecting a sequence of regions or locations and only processing the selected regions at high resolution. Reference [56] introduces an attention-based model that automatically learns to describe the content of images. Reference [57] proposed a temporal attention mechanism that allows to go beyond local temporal modeling and learns to automatically select the most relevant temporal segments given the text-generating RNN. Reference [58] proposed a question-guided spatial attention architecture that looks for regions relevant to either individual words or the entire question, repeating the process over multiple recurrent steps or ''hops.'' Reference [59] proposed a machine translation network architecture based solely on an attention mechanism, dispensing with recurrence and convolutions entirely.

III. MODEL

A. OVERVIEW OF ATTENTION GRU MODEL

The goal of attention GRU model is to learn the feature from the raw network traffic and classify malicious network traffic. Our model uses an attention network to process the raw network traffic and generate context variables which

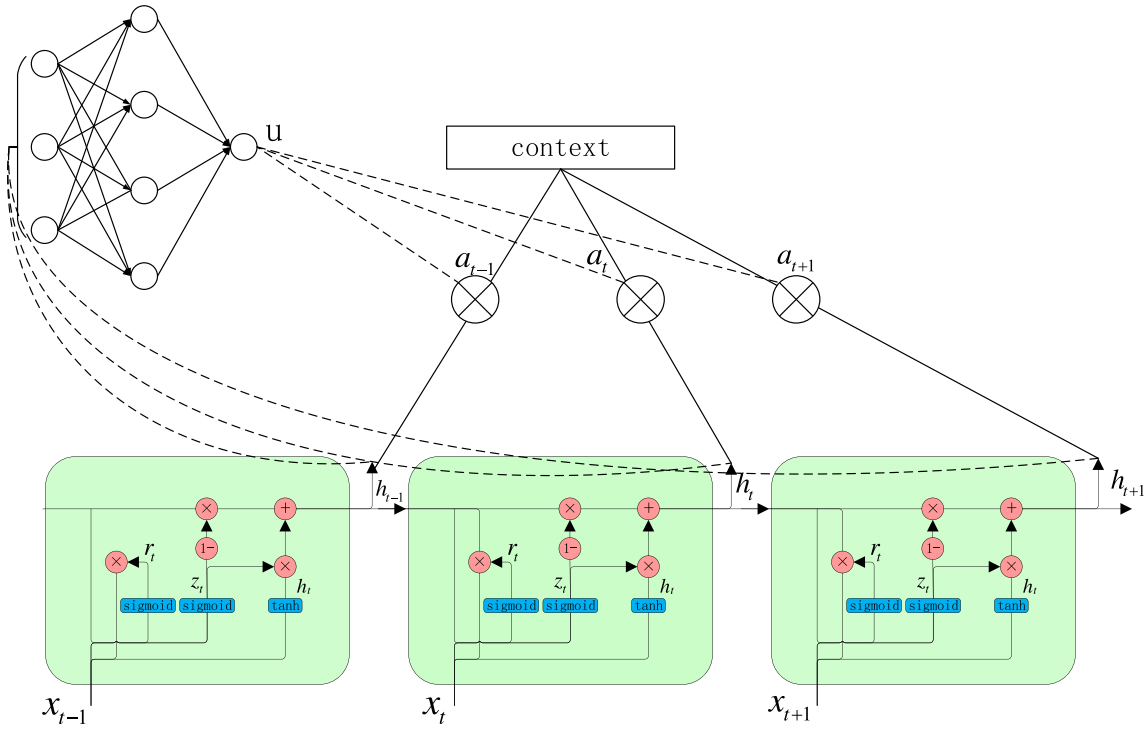


FIGURE 1. Bidirectional GRU based on attention mechanism. Hidden state h_{ij} is feed to one-layer MLP to get the weight a_{ij}.

contain all vital information of network packet. We learn from [60], [61] in modeling flows hierarchically by obtaining representations for byte and then composing those into a flow representation. The overall structure of the Model is shown in the Fig.1. It consists of five parts: a byte encoder, a byte-level attention layer, a network packet encoder, a packet-level attention layer, a classifier uses SOFTMAX functions. In the level of the packet, bytes are encoded to byte-vector and processed by byte-attention layer to extra important byte to packet. Similarly, byte-vector is encoded and processed by packet-level attention layer. Finally, the flow vector is generated and can be used as a feature of flow to classify. Our model focuses on the traffic flow level classification because many intrusion features are hidden behind the traffic flow. This study assumes that a traffic flow F has n network packet Pi and each network packet has L_i bytes B_{ij}. $B_{ij}(j \in [1, L_i])$ represent that the jth byte in the ith network packet Pi. A typical network flow is too long to process so we need to encode it and use attention mechanism to extra import network packet. The various steps of our model are listed as follows.

B. PREPROCESSING

Common flow objects, with different granularities, include TCP connections, Flows, Bidirectional flows (Biflows), Services, Hosts [62]. Our method uses the flow type which is used in the most study. We use the 5-tuple (source-IP, source-port, destination-IP, destination-port, transport level protocol) to determine one flow. Packets which have the same 5-tuple is combined to a network flow as the basic classification unit.

C. K-FOLD CROSS VALIDATION

Cross-validation is a statistical method which is used to estimate the performance of machine learning models. This method is easy to implement and results in performance estimates show that this method generally has a lower bias than other methods. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. When a specific value for k is chosen, it may be used in place of k in reference to the model. In our experiment, we determine $k = 10$, because it has good average precision of the error estimation [63]. A byte is entered as a one-hot character into the word2vector. By training the word2vector model, we get the embedding matrix W. Thus one-dimensional byte is transformed into a high-dimensional byte vector. And all the information is encoded in the flow data hence the relation between various byte can be established.

D. HIERARCHICAL ATTENTION LAYER

1) BYTE ENCODER

The byte encoding contains three steps. Firstly, two byte B_{ij} is embedded to continuous vector V_{ij} from one-hot vector by word2vector [13] $V_{ij} = W \times B_{ij}$. Secondly, V_{ij} in two directions is processed by a bidirectional GRU, and contextual information in the annotation will be incorporated. The bidirectional GRU have forward and reverse GRU to process

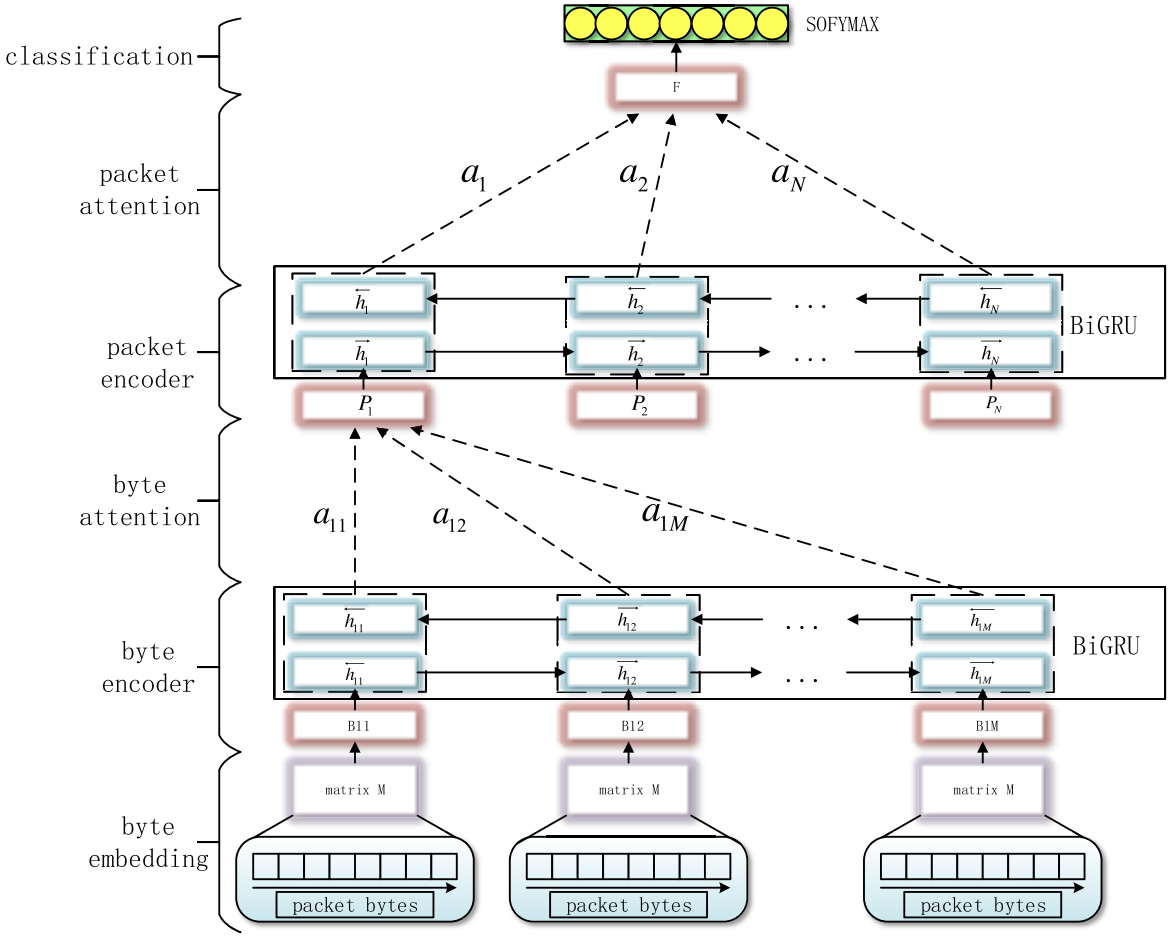


FIGURE 2. The structure of our hierarchical attention model. Our hierarchical attention model has a hierarchical structure which can build representations of bytes and packets in two levels. Moreover, it has two levels of attention mechanisms to enable it to pay attention to more important content when constructing the flow representation.

the bytes from V_{i1} to V_{iM} and from V_{iM} to V_{i1}.

$$\overrightarrow{h}_{ij} = \overrightarrow{GRU}(x_{ij}), j \in [1, M], \tag{5}$$

$$V_{ij} = W \times B_{ij}, \tag{6}$$

$$\overleftarrow{h}_{ij} = \overleftarrow{GRU}(x_{ij}), j \in [M, 1]. \tag{7}$$

From that, GRU can obtain information from forward hidden state h_{ij} and the backward hidden state h_{ij}. Then we concatenate h_{ij} and h_{ij} to get $h_{ij} = [h_{ij}, h_{ij}]$ which represent the byte w_{ij}.

2) BYTE ATTENTION

Now the feature of every V_{ij} is put into the GRU, but many bytes are not important to classify whether the packet is malicious. So we introduce the attention mechanism to assign weights to different bytes and provide some significant bytes more opportunity [64]. Attention mechanism can help to synthesize the representation of the entire packet.

$$u_{ij} = tanh(W_b \times h_{ij} + b_b), \tag{8}$$

$$a_{ij} = \frac{exp(u_{ij}{}^T u_b)}{\sum_i exp(u_{ij}^T u_b)}, \tag{9}$$

$$P_i = \sum_j a_{ij} h_{ij}. \tag{10}$$

In this method, every byte vector is assigned a specific weight a_{it}. To calculate the weight, we first feed the hidden state h_{ij} which represents the byte B_{ij} through a one-layer MLP to get u_{ij}. Then set a weight vector u_b to reflect the different importance of different bytes. This vector u_b will be optimized while training, so we get a normalized importance weight a_{ij}. After that, we can get a weighted sum of bytes vector P_i.

3) PACKET ENCODER

Just like encoding the bytes, we also use bidirectional GRU to encode the packet P_i.

$$\overrightarrow{h}_i = \overrightarrow{GRU}(P_i), i \in [1, N], \tag{11}$$

$$\overleftarrow{h}_i = \overleftarrow{GRU}(P_i), i \in [N, 1]. \tag{12}$$

Then $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ are concatenated using the same method $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$. h_i summarizes the neighbor packet around packet i but mainly focuses on the packet.

4) PACKET ATTENTION

Analogously, we assign every packet vector P_i a specific weight u_p in a similar way. This vector can measure the importance of the packet. And u_p will also be optimized through training.

$$u_i = tanh(W_b \times h_i + b_p), \tag{13}$$

$$a_i = \frac{exp(u_i{}^T u_p)}{\sum_i exp(u_i^T u_p)}, \tag{14}$$

$$F = \sum_i a_i h_i. \tag{15}$$

So we feed the packet vector h_i to the one-layer MLP and get u_i as a hidden representation. When the training and normalizing of up is complete, we get the final weight of the packet. Finally, F is the flow vector summarizes all information of the entire flow.

5) SOFTMAX CLASSIFIER

Given that flow classification is a classification task, a SOFTMAX function, which is optimized through cross-validation, is used. After getting the flow vector F, we first feed F to a one-layer MLP. Then we use the SOFTMAX function to achieve the purpose of classification.

$$V_i = SOFTMAX(W_f \times F + b_f). \tag{16}$$

where W_f is an f-dimensional vector, f is the number of attack types, and i is the ith flow. The loss function is as follows:

$$L = -sum(log(V_i)). \tag{17}$$

E. FLOW-WGAN

The problem of missing data is frequently encountered in intrusion detection studies. If we directly detect real user network traffic, it is likely to cause data breach. Inspired by WGAN, we propose a new Flow-WGAN model based on WGAN. We design this packet generator model to enhance network packet data and protect user privacy. The real original packet is processed to a fixed length. Because the length and payload of the packets are changed, we need to repair the length field in IP and TCP/UDP header, and the checksum is also recalculated. The network adapter will check the length and checksum to ensure the packet is complete and correct when it receives a packet. To accelerate the process of the model generating a packet that conforms to the packet format. We design a type and length checker referring to Table 1 and add this T(G(z)) to the generative loss function, where G(z) is the generated data and T(G(z)) is the output of the type checker. So the loss function is as follows:

$$\begin{aligned} T(\widetilde{x}) = {} & G(z)[13] - 8 + G(z)[14] + G(z)[15] - 45 \\ & + G(z)[17] - 3 + G(z)[18] - 2 \\ & + (G(z)[24] - 6) \times (G(z)[24] - 17), \end{aligned} \tag{18}$$

$$\widetilde{x} = G(x), \tag{19}$$

$$\hat{x} = \varepsilon x + (1 - \varepsilon)\widetilde{x}, \tag{20}$$

TABLE 1. Position and value in the packet.

position	field	Value(HEX)	Value(DEC)
13	IPv4	0x08	8
14	IPv4	0x08	8
15	IPv4 and Header length (20)	0x45	69
17	Total length (770)	0x03	3
18	Total length (770)	0x02	2
24	TCP	0x06	6
24	UDP	0x11	17

$$\begin{aligned} L = {} & \mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_g}[D(\widetilde{x})] - \mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_r}[D(x)] \\ & + \lambda \mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_{\widetilde{x}}}[(\|\nabla_{\widetilde{x}} D(\widetilde{x})\|_2 - 1)^2] + \mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_g}[T(\widetilde{x})]. \end{aligned} \tag{21}$$

For the discriminator training, the loss function never changes. For the generator training, the loss function is as follows.

$$min(\mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_g}[D(\widetilde{x})] + \mathop{\mathbb{E}}_{\widetilde{x}\sim\mathbb{P}_g}[T(\widetilde{x})]). \tag{22}$$

The function of T(G(z)) is feature constraint. It can limit the type and length of the generated network traffic packets. The length and type field of the generated packet will close to the limit specific value. We limit the generated packet to be TCP/UDP based on the Internet Protocol. Moreover, the length of the packet is 784 bytes, so every length field is a fixed value. Key field, corresponding position and key value are as Table 1.

IV. EXPERIMENTS

In this study, various types of experiments are done to evaluate our model.

A. DATA SETS AND PREPROCESSING

First, we use the TCPdump to capture network traffic flow in our laboratory as the training data set to generate the embedding matrix. It contains several of internet protocol such as HTTP, FTP, DNS, POP3, SMTP, SMB, telnet, et al. We use the traffic to train the unsupervised embedding model and get the embedding matrix W.

We evaluate our model with ISC-2012 [65] and ISC-2017 [66] data sets. The method of detecting in our study is designed to detect based on network flow, and our model can have a better performance than the method based on a network packet. As we have mentioned, our model will process raw data, so NSL-KDD [67] and Kyoto2009 [68] cannot be used because they contain no original data. In this section, we mainly compare the difference of performance between our model and other models.

B. ISCX-2012

Real traces are analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3,

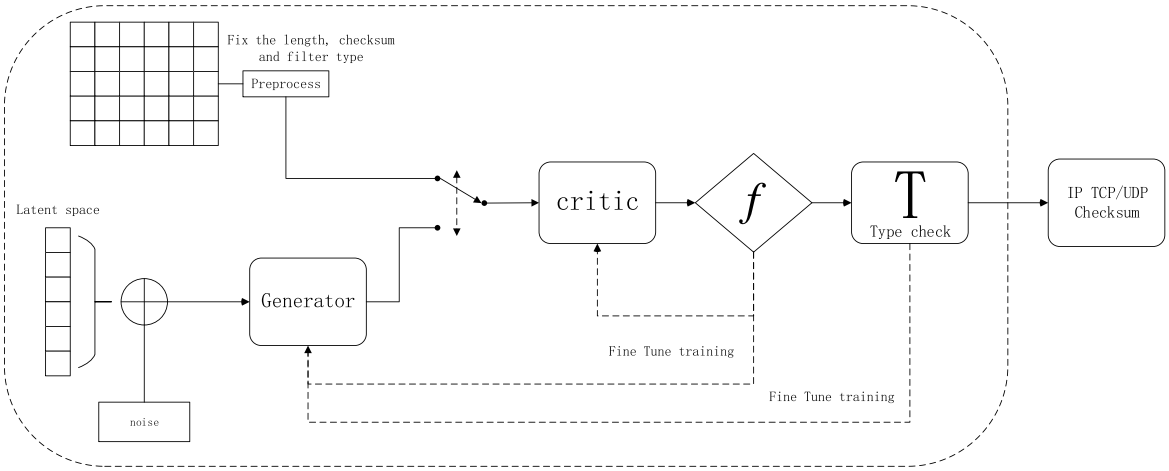


FIGURE 3. Flow-WGAN to generate new type of flow.

TABLE 2. Preprocessing results of the ISCX-2012 dataset.

	number	train	test	percentage
BFSSH	6954	4,868	2,086	1.09%
Infiltrating	10019	7,013	3,006	1.58%
HttpDoS	3395	2,377	1,019	0.53%
DDoS	21105	14,774	6,332	3.32%
normal	593,783	415,648	178,135	93.47%
sum	635256	444,679	190,577	

and FTP. Several multi-stage attacks scenarios were subsequently carried out to supply the anomalous portion of the dataset [65]. The ISCX-IDS-2012 intrusion detection evaluation dataset (https://www.unb.ca/cic/datasets/ids.html) consists of the following seven days of network activity. It contains benign network traffic and labeled four types of attack network traffic including Infiltration, BFSSH, DDoS and HttpDoS. The traffic in the ISCX-IDS-2012 data set is raw data and never be processed, so we need to classify packets into flows. The resulting statistics are shown as follows.

C. ISCX-2017

There are many malicious traffic data sets, but most of them are out of date and untrustworthy. Most of these data sets contain too few types of attacks and some of payload data is encrypted which makes the data unusable. ISCX-2017 datasets (https://www.unb.ca/cic/datasets/ids-2017.html) contains a wider variety of newer attacks, which are more suitable for simulating and studying cybersecurity in the real world [66]. This datasets contains five days generated realistic background traffic and attack traffic including Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. As we process the ISXC-2012, we classified the traffic data to flow. The resulting statistics are shown as follows.

TABLE 3. Preprocessing results of the ISCX-2017 dataset.

	total	train	test	percentage
web attack	2180	1526	654	0.79%
ddos	5326	3728	1598	1.94%
port scan	6954	4868	2086	2.53%
bot	1966	1376	590	0.71%
ssh patator	5897	4128	1769	2.14%
ftp patator	7964	5575	2389	2.89%
dos	8366	5856	2510	3.04%
normal	236506	165554	70952	85.95%
sum	275159	192611	82548	

D. PREPROCESSING FLOW

All of the data set are raw data and are non-split cap file, which needs to be classified into a network flow file. Because every traffic flow consists of many packets whose number can be huge, and a typical packet contains perhaps 1,000 or 1,500 bytes. The length of flow between any flows can be very different. Putting the data with different length to the model is difficult. So in this step, we need to process the flow and packet into the same length. We set the length of flow to M and the packet size to N. If a packet is longer than N, the exceeding part needs to be cut. If a packet is too short, the packet needs to be padded with zero. To ignore the influence of IP (Internet Protocol) address and Media Access Control (MAC) address, IP and MAC are replaced with random IP and MAC.

1) EVALUATION METRICS

There are three widely used in intrusion detection field metrics to evaluate our algorithm: accuracy (ACC), true positive rate (TPR) and false positive rate (FPR). TP is the number of malicious flow correctly identified as malicious. TN is the

number of benign flow correctly identified as benign. FP is the number of benign flow incorrectly identified as malicious. FN is the number of malicious flow incorrectly identified as benign.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \tag{23}$$

$$TPR = \frac{TP}{TP + FN}, \tag{24}$$

$$FPR = \frac{FP}{FP + TN}. \tag{25}$$

E. EXPERIMENTAL SETUP

We use the tensorflow1.4.0 (GPU version) and Keras2.0.8 as the experimental framework, on the Ubuntu18.04 64bit OS, with the Intel-CORE i7 7700k, 32GB memory and NVIDIA GTX1080Ti GPU. In the experiment, 10-fold cross-validation is used. We first split the training traffic flow data into 10 groups, then take a group randomly as the validation data set. The remaining groups are taken as the training data set. Next, our model is fit on the training data set and evaluated with the validation data set. We operate the above learning-validating process for ten turns.

In training, we use the mini-batch of 64 and use stochastic gradient descent to train the model with the momentum of 0.9. Grid search in the Keras tool is used to get the suitable learning rate. We summarize the performance of our model and other state-of-the-art models.

F. IMPACT OF THE BYTES SIZE

k bytes B_{ij} is embedded to a continuous vector in the encoder section. Embedding one byte at a time and embedding four bytes at a time is quite different. One byte is only 256 in size; however, two bytes are 65536 in size which is quite large. Bytes processing can be compared with neutral language processing. A byte can be seen as an English letter, and two or more byte can be seen as a word. Usually, word embedding has better performance, but some research [69] offers an empirical exploration of the use of character-level convolutional networks and takes an excellent result. Choosing one byte as an embedding operation input unit has an advantage that it will not destroy the original structure of the packet. However, choosing more than one byte, such as two bytes or four bytes, must consider that packet has its structure. Some neighbor bytes have a specific meaning, and they should be treated as a whole. To evaluate which method is most appropriate, we compare five sizes of bytes chosen, one byte, two bytes, three bytes, four bytes, and eight bytes. We choose these five methods because all network data is binary. Using integer power as the bytes chosen can adapt to the raw network data structure better.

We use Batch gradient descent (offline training) to train our model. The BGD uses batch gradient descent method aka the method of steepest descent [70]. The BGD class can accumulate contributions to the gradient from several mini-batches before making an update, thereby enabling it to use batches that are too large to fit in memory [71]. The BGD requires the input data is fixed length. As attention can process long input, and assign a higher weight to the critical part. Considering that the most critical part of the flow is in the head part, we set the length of the packet input size as 1500, which is the maximum length of a packet. According to the statistics, the average flow length is 78.5 packets [72]. Also, the most critical part of the packet is in the head part, so we set the flow length input as 128 packets, and the total input size of flow is 192,000.

In this experiment, we compare the three sizes of bytes chosen. Moreover, use the best bytes chosen to compare with some other state-of-the-art algorithms.

TABLE 4. Performance for the different network bytes embedding sizes.

	iscx2012			iscx2017		
embedding size	ACC	TPR	FPR	ACC	TPR	FPR
one byte	95.29%	88.53%	4.03%	92.14%	89.12%	7.37%
two bytes	**99.95%**	**99.75%**	**0.04%**	**99.45%**	**98.62%**	**0.42%**
three bytes	86.18%	73.79%	12.95%	63.30%	84.85%	40.22%
four bytes	97.78%	99.39%	2.34%	95.08%	94.37%	4.80%
eight bytes	93.12%	95.99%	7.08%	91.28%	88.53%	8.28%

Table 4 shows the experimental results. In the experiment, we compare the five sizes of bytes chosen, and it indicates that the overall best performance in this study is choosing two bytes once as the embedding operation input. A possible explanation is that two-byte aligned can mirror the data structure of the network packet. By observing network traffic packets, we find that most of the vital information is two or four bytes, and the packet is usually aligned to 2 or 4 bytes boundary. As we mentioned above, one byte is too short to be embedded. The same byte in a different position in a packet may have different meanings but will be embedded to the same vector. That may cause the attention algorithms confusion. And embedding three bytes to a vector will significantly destroy the original structure of the network data. As for four or eight bytes, the granularity of embedding is too big which lead to some bytes are treated as different bytes, but actually, they have the same meaning in the packet.

G. IMPACT OF HYPER-PARAMETERS

The learning rate is perhaps the most critical hyper-parameter [73]. The convergence speed and training results have a great influence. The smaller learning rate can ensure the stable convergence of the training, but the learning speed is slow and the training time is long. The larger learning rate can improve the convergence speed to some extent, but it is easy to lead to the model. Performance will oscillate during the training period.

From Table 5, it can be seen that when the learning rate is 0.001 or 0.01 the performance of our model is best. With the increase in the learning rate, the performance of our model gradually declines. When the interest rate exceeds

TABLE 5. Performance of our model with different learning rate.

learning rate	ACC	TPR	FPR
0.001	**99.95%**	**99.75%**	**0.04%**
0.01	**99.95%**	**99.75%**	**0.04%**
0.02	99.93%	99.62%	0.06%
0.05	95.43%	88.73%	3.48%
0.1	92.79%	87.80%	6.38%
0.2	88.92%	81.70%	9.80%
0.5	74.37%	78.20%	49.02%
1	59.72%	61.29%	49.88%

TABLE 6. Performance of our model with different word embedding dimensionality.

embedding dimensions	ACC	TPR	FPR
20	99.58%	97.69%	0.15%
50	**99.95%**	**99.75%**	**0.04%**
100	99.88%	99.34%	0.04%
200	99.59%	97.59%	0.12%
300	98.09%	89.08%	0.45%
500	96.59%	80.49%	0.52%
700	95.44%	74.97%	0.60%
1000	89.53%	59.95%	3.34%

0.5, the performance of our model declines sharply. Because of the learning rate is too large, gradient descent inadvertently increases rather than decrease the training error. So in the experiment, we set the learning rate as 0.02 and when the accuracy reaches 99% we replace the learning rate as 0.01.

We look at the embedding dimensionality as a hyper-parameter from 20 to 1000. So we make experiments to find suitable embedding dimensionality. The byte embedding dimensionality is related to the bytes size chosen. In these experiments, we chose two bytes once as the embedding operation input. From the experimental results, it can be concluded that the optimal byte embedding dimensionality is 50. And the embedding dimensionality is not a very sensitive hyper-parameter. Bytes do not contain much information, so it is not necessary to set a high dimensionality. And when the embedding dimensionality is too high, the model is prone to overfitting. However, when the embedding dimensionality is too low, the model has insufficient representation ability to represent the bytes.

H. VISUALIZATION OF ATTENTION

We visualize the attention layer both in byte-level and in packet-level in the Fig. 4,5,6,7 to show how individual bytes in the packets and individual packets contribute to the flow product. Red denotes the packet weight and blue denotes the bytes weight. Every packet is divided by blank lines, and the parts in packets which is assigned too little weight

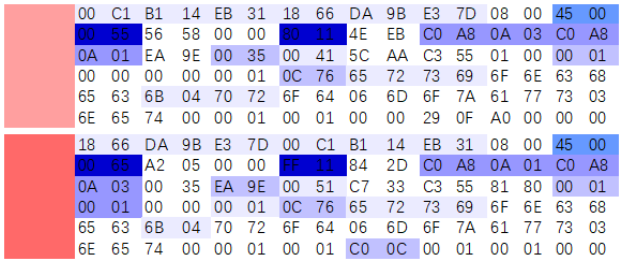


FIGURE 4. Benign.

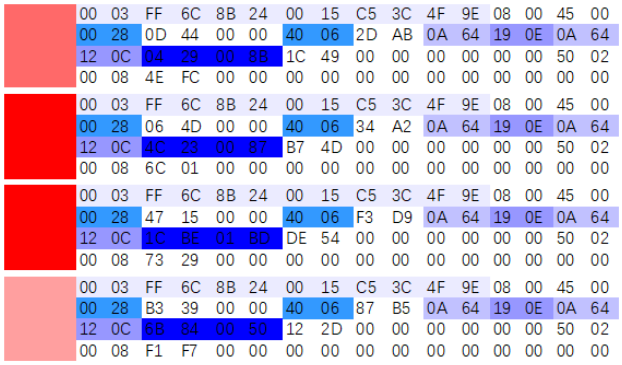


FIGURE 5. Portscan.

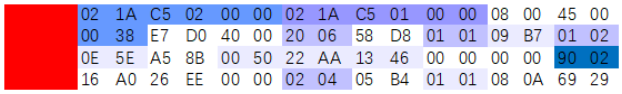


FIGURE 6. Dos.

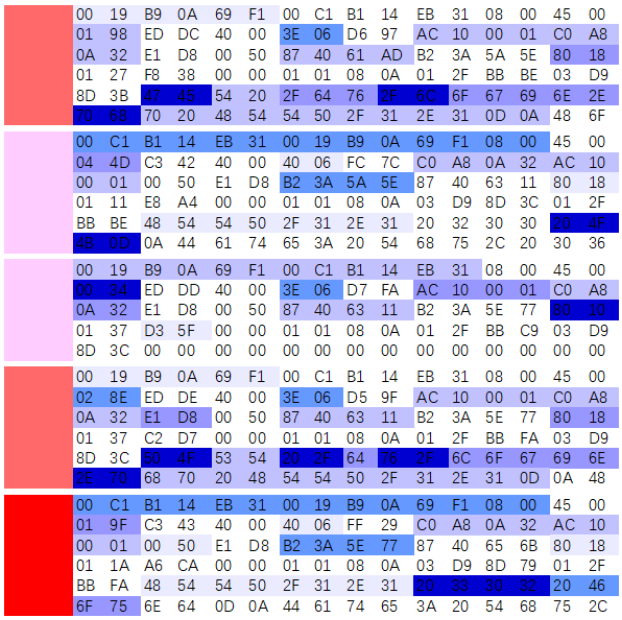


FIGURE 7. Webattack.

are omitted. Moreover, some unimportant network packets have also been omitted. Only significant bytes in significant packets play a decisive role in classification.

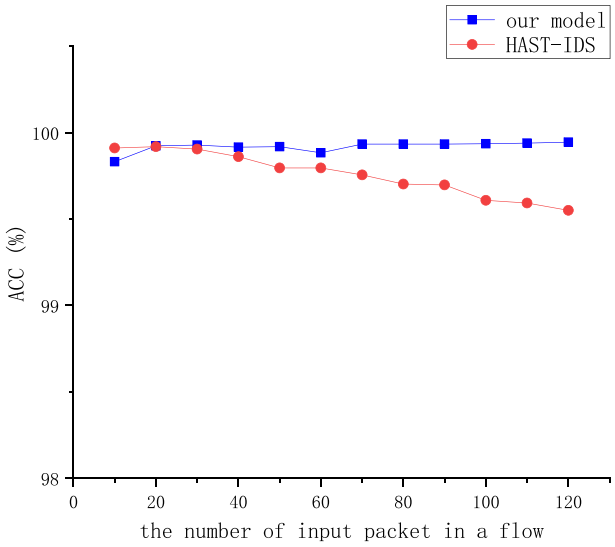


FIGURE 8. Accuracy performance with different flow length.

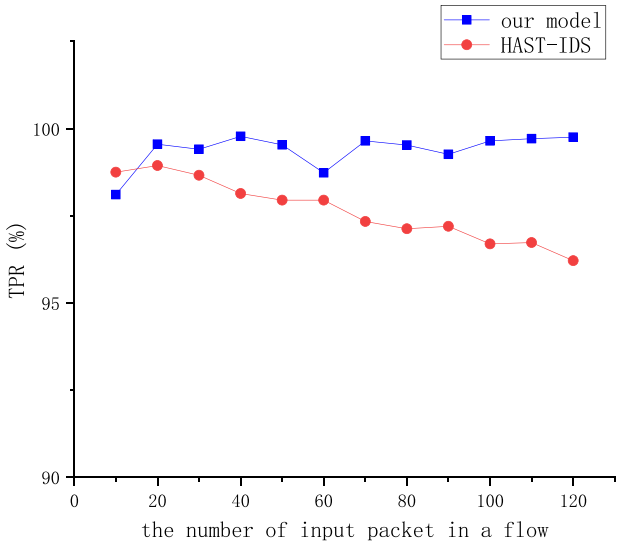


FIGURE 9. TPR performance with different flow length.

From Fig. 7 we can see, there are some critical packets are not in the head of network flow. They may be the third, fourth, even fifth package. So sometimes we cannot only process the first few packets when classifying a network flow. We must process enough traffic packets in a flow to determine this traffic flow is malicious.

Fig. 4,5,6,7 shows that our model can select critical fields in network packets, such as total length of the packet, flags, time to live and status code and any more. Plenty of zero bytes are disregarded. We can see through the visualization example of the botnet that our model can deal non-adjacent packet context. Note that our model pays attention to different bytes in different traffic flow. This result shows that our attention layer can process multiple classifications suitably.

I. LONG FLOW INPUT

Considering that most malicious traffic flow detection method based on deep learning is only using the first few packets in a flow, hackers can hide attack behavior in the payload to avoid detection. Most of malicious traffic flow detection method cannot distinguish the malicious traffic flow if the first few packets in flow do not contain any attack feature. This challenge requires the deep learning method to increase the length of network flow input. Compressing an entire long flow into a static representation may cause feature loss. However, the attention mechanism allows the model to focus on the most relevant part of the flow as needed. To evaluate these supposes, we compare our model with the most advanced model HAST-IDS with different length of the input. We set the packet length as the fixed 1500, and compare the model with different numbers of the packets as a flow input.

Fig. 8 shows that as the input packet numbers increases, the performance of our model hardly changes. When the number of input packets is less than 40, both our model and HAST-IDS have an excellent performance. However, as

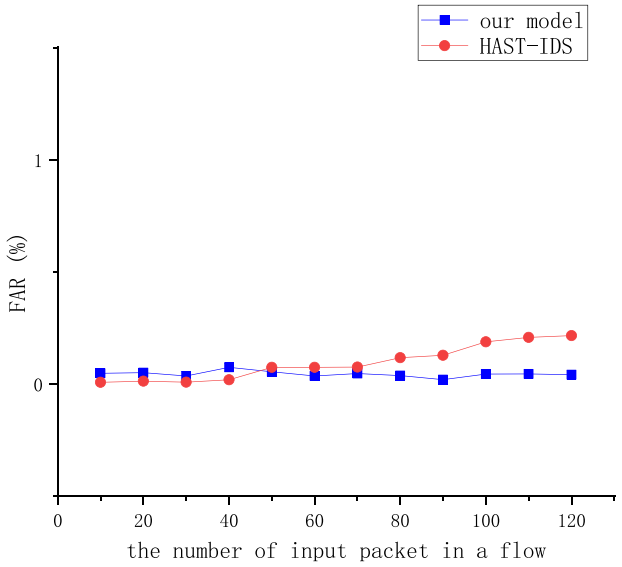


FIGURE 10. FAR performance with different flow length.

the number of input packet increase, the performance of HAST-IDS drops significantly. But our model still maintains excellent performance in ACC, TPR and FAR. This result shows that our model with attention mechanism can process well with long input of flow. In other words, if the hacker hides the attack in the payload, our model can still detect the attack signature from the payload.

J. COMPARISON WITH PUBLISHED WORK

There are many intrusion detection methods, which are based on deep learning method, have been proposed. Most of them use ISCX-2012 dataset to evaluate their method. And furthermore, some new research also evaluates their methods with the ISCX-2017 dataset. In this article, the method we proposed is running on off-line data. So all the methods

TABLE 7. Best performance of our model for ISCX-2012.

	TP	ACC	TPR	FPR
BFSSH	2043	0.9977	0.9794	0.0021
Infiltrating	2991	0.9968	0.9950	0.0032
HttpDoS	1019	0.999595966	1	0.000406206
DDoS	6323	0.9993	0.9986	0.0006
normal	176943	0.9934	0.9933	0.0054

TABLE 8. Best performance of our model for ISCX-2017.

	TP	ACC	TPR	FPR
web attack	628	99.95%	96.02%	0.02%
ddos	1566	99.92%	98.00%	0.04%
port scan	2022	99.90%	96.93%	0.03%
bot	521	99.85%	88.31%	0.07%
ssh patator	1693	99.74%	95.70%	0.17%
ftp patator	2263	99.82%	97.71%	0.12%
dos	2510	100.00%	100.00%	0.00%
normal	70596	99.18%	99.50%	2.76%

TABLE 9. Comparison with other published methods for ISCX-2012.

iscx2012	ACC	TPR	FPR
KMC+NBC	99	99.7	2.2
AMGA2-NB	94.5	92.7	7
HAST-IDS	99.89	96.96	**0.02**
RNN-RBM	99.73	97.35	**0.02**
our model	**99.95%**	**99.75%**	0.04%

we compare in this article are the off-line results. We focus on the traffic flow-level classification. This method has a better performance than the other packet-level detection methods. In this method, we compare the performance difference between our model and other deep learning methods [74], [75].

As we can see from Table 7, our model has better ACC and TPR than any other known models. This indicates that our model has reached the top-level of the state-of-the-art models. It is noteworthy that our model has an excellent performance on the DoS or DDoS attack detection. Our model can detect almost all DoS or DDoS attack and have a very low FPR.

And on the other hand, the performance of our model in detecting bot attack is not satisfactory. For the ISCX-2017 dataset, the detecting rate is only 88.31%, which is significantly lower than other types of attacks. In addition to this type, our model has quite stable performance in other types of attack detection. The detection rate for the other six types of samples all exceeds 95%. As for the ISCX-2012 dataset, the accuracy and TPR of our model are quite excellent. The reason may be that the ISCX-2012 dataset has less diversity and types of attack.

TABLE 10. Table comparison with HSAT-IDS for generated packets.

data type	original packets		generated packets	
model	our model	HAST-IDS	our model	HAST-IDS
false acceptance	304	**276**	**36**	127
false acceptance rate	0.30%	**0.28%**	**0.26%**	0.91%
normal	99696	99724	13964	13873
percentage	99.70%	99.72%	99.74%	99.09%
total	100000	100000	14000	14000

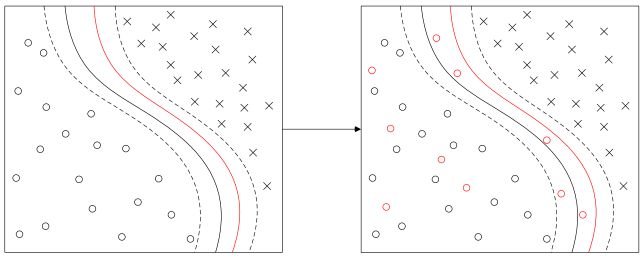


FIGURE 11. Influence of generated data. The circles indicate benign data, and the forks indicate abnormal data. The red circles indicate the new data generated with Flow-GAN. The solid black line indicates the classification result of other classifiers, and the red solid line indicates the classification result of our classification model.

K. GENERATING PACKET WITH FLOW-WGAN

We first use the WGAN-GP to generate packets, but it is very difficult to convergence for the training. And it is also very difficult to verify that the generated traffic packets meet the format. So it is essential to ensure the accuracy of the traffic packet format. So we add the type and length checker to improve the training process. After using the new loss function, the training process has become faster, and the proportion of correct traffic packets has increased. 20000 real packets are used to train the GAN model, and 12000 packets are generated to test how our model deals with a new type of Internet application. Every day, there are many new applications which are created on the Internet, and we expect our model can correctly classify these applications, rather than detecting them as malicious traffic.

To evaluate the ability of our model classifying new Internet application, 20000 real packets and 12000 generated packets are feed into our model and HAST-IDS. Table 10 shows that when processing the generated packets, classifiers have a higher FAR than processing the real packets. This result shows that Flow-WGAN dose generates different types of applications which can be used to evaluate an intrusion detection method. One possible explanation is that Flow-WGAN extracts different information because it uses different networks comparing with the classifier model. Fig.6 shows as the classifying pseudo-hyperplane and the data points are divided into two classes: benign (circles) and malicious (fork). The class of the points between the two pseudo-hyperplanes is difficult to determine. We find that Flow-WGAN can be used to help to improve or evaluate a classifier. Fig. 11 shows that Flow-WGAN can

generate points between the two pseudo-hyperplanes like other imputation methods. Moreover, comparing with HAST-IDS, we find that our model performs better when process a new type application outside the training set. This indicates the attention mechanism can help model learning the key features and reducing over-fitting for the training set.

V. CONCLUSION

In order to avoid hand-designing accurate traffic features, we proposed the hierarchical attention language-based model. It uses the attention mechanism and bidirectional GRU to learn hierarchical features from the raw network data. Our model can use the raw packet as input at the byte-level. It progressively builds a flow vector by aggregating critical bytes into packet vectors and then aggregating critical packets vectors to flow vectors. The performance of the proposed model is analyzed by comparing with other published methods. The comparison of the results demonstrated that the proposed model exhibits higher accuracy, less FDR, and is suitable for improving malicious network traffic detection. In addition, the training time of our model is 30% less than the training time of HAST-IDS. This indicated our model can find the critical parts with attention mechanism and convergence faster.

Our work proposed a hierarchical attention model which can learn information from two levels of the network flow structure and can represent the bytes better. We used the bytes embedding method to encode the bytes, so the angle between two bytes vectors can represent the relationship of them. We designed a Flow-WGAN to generate a new type of network traffic to evaluate the performance of the classifiers or improve the classifiers.

ACKNOWLEDGMENTS

The authors would like to thank Xiaozhou Ye, Lei Liu, Lei Song, Jinghong Wu, Ce Zheng, Yiran Hao, Chuanhong Li, Liyi Kang, Jun Yang for insightful comments.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016.
[2] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of Data Mining in Computer Security*. Boston, MA, USA: Springer, 2002, pp. 77–101.
[3] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proc. IEEE 19th Annu. Comput. Security Appl. Conf.*, Dec. 2003, pp. 14–23.
[4] G. Poojitha, K. N. Kumar, and P. J. Reddy, "Intrusion detection using artificial neural network," in *Proc. IEEE 2nd Int. Conf. Comput., Commun. Netw. Technol.*, Jul. 2010, pp. 1–7.
[5] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Comput. Secur.*, vol. 77, pp. 304–314, Aug. 2018.
[6] M. Montazeri, M. Montazeri, H. R. Naji, and A. Faraahi, "A novel memetic feature selection algorithm," in *Proc. IEEE 5th Conf. Inf. Knowl. Technol.*, May 2013, pp. 295–300.
[7] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[8] B. J. Radford, B. D. Richardson, and S. E. Davis, "Sequence aggregation rules for anomaly detection in computer network traffic," 2018, *arXiv:1805.03735*. [Online]. Available: https://arxiv.org/abs/1805.03735
[9] Y. Hao, Y. Sheng, J. Wang, and C. Li, "Network security event prediction based on recurrent neural network (in Chinese)," *J. Netw. New Media*, vol. 6, no. 5, pp. 54–58, 2017.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: https://arxiv.org/abs/1412.3555
[12] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2010, pp. 384–394.
[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[14] R. Socher, J. Bauer, A. Y. Ng, and C. D. Manning, "Parsing with compositional vector grammars," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, Aug. 2013, pp. 455–465.
[15] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Language Process.*, Oct. 2013, pp. 1631–1642.
[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2013, pp. 3111–3119.
[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: https://arxiv.org/abs/1409.0473
[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.
[19] Z. Zhou, F. Xiong, X. Chen, Y. He, and S. Mumtaz, "Energy-efficient vehicular heterogeneous networks for green cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1522–1531, Apr. 2018.
[20] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access control and resource allocation for M2M communications in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3093–3103, May 2019.
[21] Z. Zhou, W. Bingchen, D. Mianxiong, and K. Ota, "Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.
[22] V. Jyothsna, V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *Int. J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, 2011.
[23] 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.
[24] 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.
[25] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[26] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Netw.*, vol. 32, no. 4, pp. 54–60, Jul./Aug. 2018.
[27] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 160–167.
[28] 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.
[29] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*. Cham, Switzerland: Springer, 2016, pp. 137–149.
[30] 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)*, Jul. 2017, pp. 43–48.

[31] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Comput. Secur.*, vol. 43, no. 6, pp. 1–18, 2014.

[32] G. Loganathan, J. Samarabandu, and X. Wang, "Real-time intrusion detection in network traffic using adaptive and auto-scaling stream processor," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[33] L. Nicholas, S. Y. Ooi, Y. H. Pang, S. O. Hwang, and S.-Y. Tan, "Study of long short-term memory in flow-based network intrusion detection system," *J. Intell. Fuzzy Syst.*, vol. 35, no. 6, pp. 5947–5957, Dec. 2018.

[34] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[35] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *Proc. IEEE Int. Conf. Acoust., Speech Signal (ICASSP)*, Mar. 2017, pp. 2482–2486.

[36] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, Apr. 2015.

[37] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Inf. Sci.*, vol. 294, pp. 95–108, Feb. 2015.

[38] 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.

[39] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," 2018, *arXiv:1803.10769*. [Online]. Available: https://arxiv.org/abs/1803.10769

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[41] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[42] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5077–5086.

[43] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: https://arxiv.org/abs/1511.06434

[44] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," 2016, *arXiv:1609.03126*. [Online]. Available: https://arxiv.org/abs/1609.03126

[45] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1125–1134.

[46] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1857–1865.

[47] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2223–2232.

[48] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4681–4690.

[49] T. Kim, B. Kim, M. Cha, and J. Kim, "Unsupervised visual attribute transfer with reconfigurable generative adversarial networks," 2017, *arXiv:1707.09798*. [Online]. Available: https://arxiv.org/abs/1707.09798

[50] M. Li, W. Zuo, and D. Zhang, "Deep identity-aware transfer of facial attributes," 2016, *arXiv:1610.05586*. [Online]. Available: https://arxiv.org/abs/1610.05586

[51] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4030–4038.

[52] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5810–5818.

[53] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: https://arxiv.org/abs/1406.1078

[54] R. A. Rensink, "The dynamic representation of scenes," *Vis. Cognit.*, vol. 7, nos. 1–3, pp. 17–42, 2000.

[55] V. Mnih, N. Heess, K. Kavukcuoglu, and A. Graves, "Recurrent models of visual attention," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2204–2212.

[56] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," 2015, *arXiv:1502.03044*. [Online]. Available: https://arxiv.org/abs/1502.03044

[57] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4507–4515.

[58] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *Proc. Eur. Conf. Comput. Vis.*. Cham, Switzerland: Springer, 2016, pp. 451–466.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[60] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*. [Online]. Available: https://arxiv.org/abs/1508.04025

[61] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1480–1489.

[62] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan./Feb. 2012.

[63] 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.

[64] Q. Zhou and H. Wu, "NLP at IEST 2018: BiLSTM-attention and LSTM-attention via soft voting in emotion classification," in *Proc. 9th Workshop Comput. Approaches Subjectivity, Sentiment Social Media Anal.*, Oct. 2018, pp. 189–194.

[65] 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, 2012.

[66] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, Jan. 2018, pp. 108–116.

[67] 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.*, Jul. 2009, pp. 1–6.

[68] J. Song, H. Takakura, and Y. Okabe. (2006). *Description of Kyoto University Benchmark Data*. Accessed: Mar. 15 2016. [Online]. Available: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf

[69] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.

[70] P. J. W. Debye, *The Collected Papers of Peter J., W. Debye*. New York, NY, USA: Interscience, 1954.

[71] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: A machine learning research library," 2013, *arXiv:1308.4214*. [Online]. Available: https://arxiv.org/abs/1308.4214

[72] P. Jurkiewicz, G. Rzym, and P. Boryło, "Flow length and size distributions in campus Internet traffic," 2018, *arXiv:1809.03486*. [Online]. Available: https://arxiv.org/abs/1809.03486

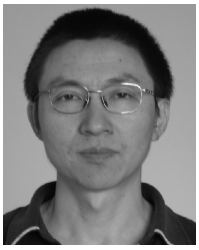
[73] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (Adaptive Computation and Machine Learning Series). London, U.K.: Chapman & Hall, 2018.

[74] 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. Inf. (ICOCI)*, vol. 49, Aug. 2013, pp. 28–30.

[75] G. Kumar and K. Kumar, "Design of an evolutionary approach for intrusion detection," *Sci. World J.*, vol. 2013, Sep. 2013, Art. no. 962185.

LUCHAO HAN received the bachelor's degree from North China Electric Power University, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree with the National Network New Media Engineering Research Center, Chinese Academy of Sciences, Beijing. His current research interests include network intrusion detection, cybersecurity, and big data.

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 was with Sanyo Electric Company Ltd., Japan, from 2003 to 2008, and Nokia, Japan, from 2008 to 2010. He conducted research on communications and integrated systems at Osaka University, from 2010 to 2011, and the Tokyo Institute of Technology, from 2011 to 2014. He is currently an Academic Researcher with the Chinese Academy of Sciences. His current research interests include machine learning, distributed big data, the future Internet, intelligence-enabled networking, multimedia service, and optimization algorithms.



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 currently a Research Fellow with IACAS and a Professor with the Chinese Academy of Sciences. His research interests include multimedia communication and pattern recognition.