Based on deep-learning apporach to real-time identify traffic of VoIP applications

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—With their high service quality and low price cost, VoIP applications win most of the users' favor. However, owing to the convenience of VoIP service, there are some social tragedies caused by fraud call. In order to make VoIP applications serve human better, it is important to keep VoIP applications under supervision. The upgrade of VoIP technology makes the traditional identification method inefficient, so it is more difficult to manage VoIP applications. In this paper, we propose a generic approach, which can identify VoIP traffic from all kinds of VoIP applications (non-encrypted and encrypted). Using this approach, we can also determine source application of captured VoIP traffic in real time. Our approach uses deep learning to extract statistical features for matrix which is composed of several continuous RTP or RTCP packets. It is able to find most useful features with deep learning approach, in the meantime, it can get rid of human effort in exploring features. In addition, we design a real-time identification system to verify the efficiency of our approach. The evaluation results show that our approach can identify VoIP applications accurately.**

***Index Terms*—VoIP, traffic identification, deep learning, real-time.**

I. Introduction

Recently, with their price advantage, VoIP (Voice over Internet Protocol) applications have been used more and more among the people. In some private networks, network traffic generated by VoIP applications need to be controlled. It is necessary to identify VoIP application traffic accurately. VoIP applications can provide many convenient services, like anonymous service, multi-line calls. Because of the convenience of these services, VoIP applications are used for fraud by some unruly elements. So determine VoIP traffic comes from which application can help us deal with emergencies.

However, the upgrading of VoIP technology makes identification task more difficult. In most of VoIP applications, SIP or H.323 protocol is used in call connection phase, RTP/RTCP protocol is used in voice transmission phase. Encryption technologies such as SSL/TLS, SIPS, WEP and WAP/WAP2 may be used to encrypt call connections, SRTP/SRTCP may be used to encrypt voice transmission. Identifying VoIP traffic is getting harder and harder. It is urgent to find a more effective and more generic way for identifying VoIP traffic.

In this paper, our main goal is finding a way to determine VoIP traffic's original application. It is impossible to identify traffic using call signaling which is encrypted using SSL/TLS, WEP and WAP/WAP2 in call connection phase. We adopt a method that only relies on the traffic generated during the voice transmission phase. Deep Learning provides us with ideas to solve the this problem. The features extracted by deep learning are not only more reliable than the features extracted by humans, but also greatly improve the identification efficiency.

The concept of real time mentioned in this article is relative, if we can identify the source application of a call flow in an acceptable time, we think it is real time. Many researchers identify VoIP applications by extracting the characteristics of bidirectional voice transmission stream, it is not feasible for real-time identification. Therefore, we proposed a method to extract features for multiple continuous packets generated within seconds or even a few milliseconds.

The remainder of this paper is structured as follows. In section II, we describes the previously published related work. In section III, there is a brief introduction to several methods used in this paper. Then we present the architecture of our real-time identification system in section IV. In sections V, VI and VII, we describe the dataset and model used for deep learning, and how to perform real-time identification. The performance evaluation is shown in section VIII. The conclusion is shown in section IX.

II. Related Work

Khan, F. I. U. A. (2008) [1] proposed a generic technique for VoIP traffic detection. They analyze the characteristics of packet-inter arrival time, packet size and rate of packet exchange in the whole voice transmission stream. Their main goal is to use generic features to distinguish between VoIP traffic and other types of traffic. The following conclusions are given in this paper: Average packets/Sec rate is greater in VoIP as compared to other applications; Average packet size in bytes is small in VoIP as compared to other applications. Their method is important in detecting VoIP traffic, but it can not be used for identify specific VoIP applications. Our main goal is to further identify specific original application for VoIP traffic.

Yildirim, T., & Radcliffe, P. J. (2010, August) [2] proposed a method that can detect VoIP traffic quickly, this method is aiming at improving VoIP service quality. Packets whose length in range of 60 to 150 bytes are treated as VoIP packets. The article is mainly aimed at improving the quality of VoIP services, so detecting efficiency has a higher priority than accuracy. This method can not be used for identify specific VoIP applications either, and the length of traffic generated by latest VoIP application is no longer limited to 60-150 bytes. In our research, the length of packets of several applications is more than 200. But it still is a simple and effective way to help us extract VoIP traffic in all kinds of network traffic.

Gomes, J. V., Inácio, P. R., Pereira, M., Freire, M. M., & Monteiro, P. P. (2013) [3] proposed a method that identifies Peer-to-Peer VoIP sessions using entropy and codec properties. The mechanism classifies the VoIP flows, in real-time, based on the speech codec used in the session. Classifier does not use the lengths of the packets individually, it explores their level of heterogeneity using entropy to emphasize such feature. Different codecs used by different VoIP applications are listed, and the lengths of the payloads and the entropy of the first three minutes of VoIP sessions using different CBR or VBR codecs are drawn in this paper. The results of the performance evaluation shows this method can identify VoIP sessions accurately. The proposed method is able to identify VoIP applications in real-time, it is similar to our goal. 3 minutes are still too long to identify VoIP traffic for our requirement. We want to improve the effectiveness and accuracy using deep learning.

Alshammari, R., & Zincir-Heywood, A. N. (2015) [4] structure feature sets for traffic excluding payload. The traffic is complete bidirectional flow between client and server. In this paper, fiat(forward inter-arriaval time), biat(backward inter-arriaval time), fpkt(forward packet length), bpkt(backward packet length), proto and duration etc. are listed. And researchers are trying to train dataset using 3 supervised learning methods, C5.0, Ada Boost and Genetic Programming. The results show C5.0 is better than the others, for Skype, it achieves 100% DR, the other VoIP application achieves more than 90% too. This paper lists so many features for complete voice flow, we want to collect more features just for a short voice flow using deep learning, so that classifier can be more accurate and effective.

Zhanyi Wang [5] proposed the application of Deep Learning on traffic identification. They think it is difficult and time-consuming to find the features for traffic identification. They propose a method based on neural network and deep learning, and they apply the method to protocol classification and unknown protocol identification, all of the results show the deep learning method works very well on protocol classification. Their method provide us a idea to identify application of VoIP traffic in real time.

There are several differences between our method and above methods: 1) we use deep learning to extract feature set. The feature set is more credible than extracted by human; 2) IP and UDP header are not included in our dataset. The reason why we do not use them is the development of NAT technology and network proxy technology make the identification based on IP address limited and VoIP applications always use non-specific port; 3) we only use a limited number of packets to identify application. This is the basis of our real-time identification. Based on these differences, our method has several advantages, automatic generation of feature set, generic for non-encrypted and encrypted VoIP applications, used for real-time identification.

III. Methodology

In order to identify a flow's original application, we need to extract the features of the traffic generated at the beginning of the VoIP call. Most of the VoIP applications use RTP/RTCP protocol to transfer data, the ability to identify a single packet is low. In this paper, we extract features for a object (we use k-packet to denote it in the following) which consists of several continuous VoIP packets. According to our investigation, the number of packets generated by VoIP applications can reach tens or even hundreds in a second, which proves our method is feasible. We did some researches for 10 VoIP applications. According to our analysis, there are 8 VoIP applications (AltCall, Jumblo, Xlite, Zoiper UUCall, Eyebeam, ExpressTalk and Bria) using constant-length packets for transmission. 2 VoIP applications (Skype and KCCall) use variable-length packets. The length number used by Skype is completely random. About KCCall, it only limits to 2 values. Most of them use single payload type except KCCall which uses 2 types. The above analysis can divide 10 VoIP applications into several classes, our goal is to use deep learning techniques to extract as many features as possible for k-packet, including basic features such as length of packet, type of payload, and other features can not be observed by humans. The features learned by deep learning can classify them more detailed.

We apply supervised learning method to train the dataset. Constructing and labeling dataset is troublesome for researchers. In order to get raw VoIP traffic, we deploy several computers installed with VoIP applications in real network. To label them accurately, we adopt some traffic capture tools which are process-based. They can help us get pure raw traffic for each application, so we can label traffic using the name of VoIP application. Next problem is how to construct our training dataset with raw traffic data and ensure the dataset is uniform as soon as possible. We design a way that uses sliding window to select k-packet into training dataset, the next step is randomized in [1, k]. In Section V, we will introduce how to construct dataset detailed.

In order to select a suitable deep learning model for our training phase, we compared several classic deep neural network models. We considered factors such as hardware requirements and time-consuming and so on, we selected AlexNet to training our dataset finally. The training is carried out on a machine with 24G RAM and a quad-core Intel processor of 3.6GHz. The machine is equipped with NVIDIA GeForce GTX 1070(8GB) to accelerate computing. CUDA Toolkit 8.0 is used in our experiment to support parallel computing.

Collecting raw traffic data, constructing and labeling dataset and training dataset are offline, they are the basis for identifying traffic online in real time. In real-time identification phase, we parse the {srcIP, dstIP, srcPort, dstPort} in IP and UDP header, the quad-tuple will be used to separate flows as a key. We generate a k-packet for each flow with their specific key, then we identify the k-packet with the offline identification model. For each flow with specific key, the identification result will be saved as the value of the key. The {key, value} is saved into our database, we can identify the following traffic using their quad-tuple.

IV. Architecture

In this section, we introduce the architecture of real-time identification system. There are 2 main phases in our architecture, offline training and online identification. We introduce the architecture of these two phases separately.

A. *Offline Training*

In offline training phase, we use 10 kinds of traffic of captured VoIP applications training deep leaning model, our goal is to maximize the accuracy of training and validation data. There are mainly 2 procedures, data preprocessing and model training. In preprocessing, we label our dataset firstly, random select k-packet with sliding window secondly. In the procedure of converting k-packets into matrices, we remove IP header and UDP header and normalize them. Then we input these matrices and label vector into deep learning model. The training process is maximizing the loss function to find an optimal function for identification. The output of training process is the identification model that we will use in real-time identification. In evaluation procedure, We evaluate the identification model by calculating loss and accuracy for evaluation data which needs to be normalized too. Offline training detailed in Figure 1.

B. *Online Identification*

Online identification is the final part of our system. In this phase, there are 2 operations for every packet. We parse IP address and UDP port as its key for every packet. We maintain

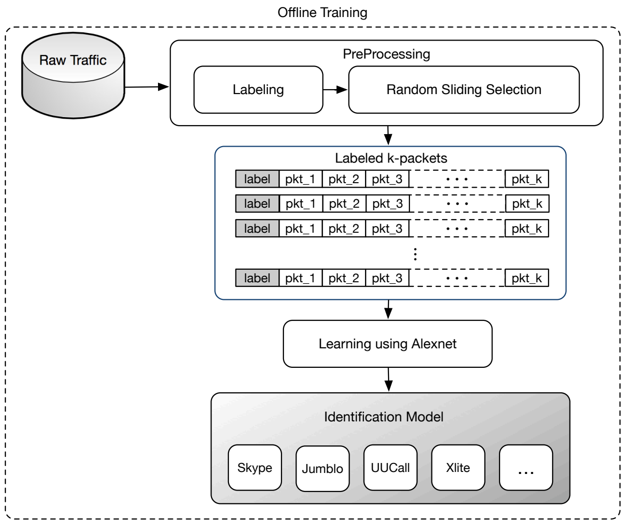

Fig. 1. Architecture of offline training.

a map database for these keys. If the value of the corresponding key has been set, the value is the identification result of the packet. If the value is null, we search the key in pending flows database and add this packet into corresponding pending flow. When a pending flow is ready to be identified, it will be identified by identification model and set the identification result in map database. In Figure 2, the number of pending flow key_3 reaches k, it is ready to be identified. We show 2 captured packets whose keys are key_1 and key_2, packet corresponding to key_1 will be identify with map database in real time, packet corresponding to key_2 will be appended into its pending flow.

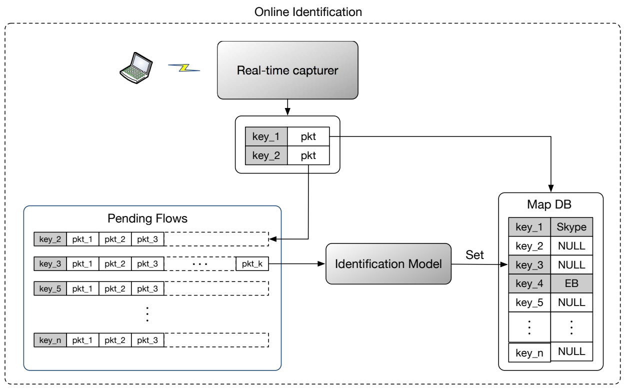

Fig. 2. Architecture of online identification.

V. Traffic Collection and Processing

Traffic collection is the beginning of all of the works. In order to get raw traffic of 10 VoIP applications mentioned above, we deploy 8 computers installed 10 kinds of VoIP applications. We installed 6 computers with Windows operating system, the others 2 computers installed with Linux operating system, because there are only 4 applications can be installed on Linux platform. We can establish ordinary VoIP calls among them, and all of them can dial the phone number directly. There are 4 cell phones ready to be called.

For the purpose of labeling data easier, we use the process-based capture tool QPA under Windows operating system; We use tcpdump to capture traffic based on port, which can be known after process occupied. Additional analysis is done using the well-known Wireshark network packet analysis tool. QPA provide GUI application under Windows OS, it is easy to use for capturing traffic for a single process. Under Linux platform, we need to get pid of a process, then we use command like "tcpdump -i eth_name trans_protocol port pid" to capture traffic of a specific process. Because of process-based and port-based capture, it is easy to label raw traffic with their process name.

We need to construct our training dataset with captured raw traffic. We process the raw traffic to k-packets, we use sliding window to select k-packet into our dataset. The length of the window is k, and the next step is random. It ensures that our datasets will not be overly similar or overly vague. Figure 3 shows how we select 4 k-packets for $k = 10$. In this paper, a variety of identification models are trained according to different k, including 2, 4, 6, 8, 10, 20, 40 and 100. We have 3 steps to convert a k-packet to matrix. 1) Remove IP and UDP header; 2) Convert the k-packet to matrix on the basis of ASCII code; 3) Normalize the matrix. The length of VoIP packets is generally small around 50$\tilde{2}$10, so we set column of the matrix to 256. The row of the matrix is k, indicating a k-packet includes k packets.

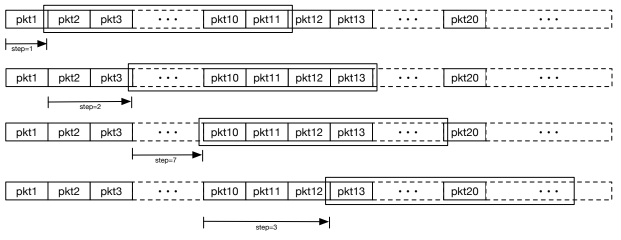

Fig. 3. How to process raw traffic to get k-packets(k=10).

VI. Learning using Deep-Learning model

A. Deep Learning model

With the rise of artificial intelligence technology, more and more excellent deep learning models appear. The oldest is CNN (Convolutional Neural Network) proposed by [6]. The LeNet is a development of the CNN proposed by [7]. [8] published an incredible neural network called AlexNet, this model get the best performance in the ImageNet competition held in 2012. [9] proposed VGG. The GoogleNet make a daring attempt in the design of the network, which proposed by [10]. [11] proposed ResNet, which has a deeper network structure up to 152 layers.

We compare CNN, AlexNet and GoogleNet with k set to 20 using a small part of dataset. The accuracy of CNN is above 94%, the accuracies of AlexNet and GoogleNet are close to 1. GoogleNet costs more time and space than the other 2 models. We did comprehensive consideration about accuracy, efficiency, hardware requirements, data set size and other conditions, we decide to use AlexNet in the next further training. With the increase of dataset, we can use the models with a deeper network structure such as GoogleNet and ResNet.

B. AlexNet

AlexNet includes 8 layers, consists of 5 convolution layers and 3 fully connected layers. AlexNet adopts relu (Rectified Linear Units) function as its activation function, and it applies dropout in learning phase. These 2 ideas shortens the training cycle so that it increases efficiency.

The inputs of AlexNet are matrices with size of k×l, where k is number of packets used for k-packet, l is the column of matrix, we use 256. In our experiment, several k values we select are not fit into AlexNet. So we do some adjustment on original AlexNet, we adjust the filter kernel to 5×5, and we change the padding type to same in each layer. The first 7 layers in the structure dropout the neural according to a probability of 0.5, the first 2 fully connected layers are activated using the relu activation function, and the last fully connected layer is activated using softmax function. We only keep a max-pooling operation in the last convolution layer, the pooling window set to 2×2.

C. Learning

We use softmax function as the activation function of the last layer. There are 10 types of VoIP traffic in our experiment, we calculate the posterior probability for unknown VoIP traffic. Softmax function shows as

$$\hat{P}(V_i) = softmax(V_i) = \frac{e^{V_i}}{\sum\limits_{j=1}^{n} e^{V_j}} \tag{1}$$

where $(V_1, V_2, \cdots, V_i, \cdots, V_n)$ denotes the matrix to be inputted into softmax in the 8th layer; $(\hat{P}(V_1), \hat{P}(V_2), \cdots, \hat{P}(V_i), \cdots, \hat{P}(V_n))$ is the output of the 8th layer, it is posterior probability distribution of k-packet, it is a vector with size of $1 \times n$; n is 10 in our experiment, it denotes the number of training VoIP application classes.

During the training, we train data to find the most optimal weights to maximize the likelihood estimate of k-packet. To minimize the CCE (categorical cross-entropy), we have the formula as follow,

$$CCE(V_i) = -\sum_{i=1}^{n} P(V_i) \times \log(\hat{P}(V_i)) \tag{2}$$

where $P(V_i)$ denotes the real probability V_i, it is the target matrix generated according to the labels corresponding to the training data. $P(V_j) = 1$, where j is the label corresponding to VoIP application that this k-packet belongs.$P(V_i) = 0$,where $i \neq j$.

We use SGD (Stochastic Gradient Descent) optimizer to minimize the loss function in this paper, and apply nesterov momentum to update gradient after every iteration. The formula of gradient increment shown as follow,

$$\Delta X_t = \tau M_{t-1} - \eta \nabla f(X_{t-1} + \tau M_{t-1}) \tag{3}$$

where τ denotes the factor of momentum, η denotes learning rate. $g_t = \nabla f(X_{t-1} + \tau M_{t-1})$ is the gradient of transition point, where ΔX_t denotes the real decreasing displacement, X_t denotes the position at time t, M_t denotes the momentum ate time t.

In our experiment, learning rate will decay after each epoch, decay rule shown as:

$$\eta_i = \eta_{i-1} \times \frac{1}{1 + \rho \times i} \tag{4}$$

where ρ denotes the decay factor, i denotes the number of epoch.

D. *Feature Set*

Our method is to get a more precise feature set for identifying VoIP traffic. The feature set extracted by deep learning is unreadable. We use these feature set to train SVM, random forest, decision tree and naive bayes. With this feature set, they all achieved good accuracy. We compare the accuracy of using our feature set and the feature set used in some previous research in Section VIII-C.

VII. Real-time identification

In this section, we introduce how to real-time identify VoIP applications with identification model. To capture the VoIP traffic in real time, we design a real-time traffic capturer. This capturer can filter RTP/RTCP traffic and separate flows by IP address and UDP port.

In most of VoIP applications, H.323 and SIP protocol are used for establishing and ending VoIP call. They both support call hold, call transfer, call forwarding, call waiting and some other supplementary services. So there are some researchers devote to monitor VoIP traffic based on call control signaling. It is no longer feasible to monitor VoIP traffic based on call control signaling. Here are several reasons: 1) SIP and H.323 protocols have their own development, and they support both UDP and TCP now. 2) The usage of ports is more and more irregular, it makes many identification methods useless. 3) More and more VoIP applications are encrypted to the traffic of call control. 4) For some VoIP applications, the server used for communication and that used for transmitting call control signaling are different. Due to these above reasons, it is hard to monitor VoIP call control signaling. These reasons also are our starting point to identify VoIP traffic with RTP/RTCP packets, all applications use RTP/RTCP protocol to transfer voice data.

In order to detect VoIP voice traffic in real time, we have some rules. We only monitor the packets whose transport layer's protocol is UDP, and their packet length is less than 256. We parse the UDP packet that meets our requirements by RTP or RTCP protocol. If the UDP packet can be parsed correctly, we will process this packet with 2 operations, append it into corresponding pending flow or search identification result in map database. If it can not be parsed, we ignored it. Moreover, we ignore packet that is alone, it makes no sense to monitor it. It is easy but it works better than the way based on call control signaling.

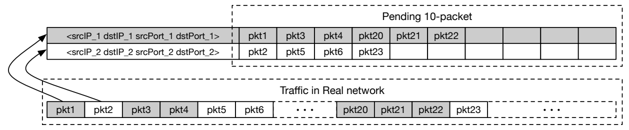

Fig. 4. To separate flows by IP address and UDP port in real network.

The capturer can separate pending flows and map packet into map database using their IP address and UDP port. Figure 4 shows the details to separate flows by IP address and UDP port. If a packet can not find the corresponding pending flow, we create a new pending flow and use its IP address and UDP port as key. If a packet is appended into pending flow, and the number of packet reaches k, we process k-packet to matrix. The rules are similar with above. We skip it here. We input the matrix into identification model, it will output a vector whose size is $1 \times n$, n denotes the number of VoIP applications that our identification model can identify. The vector indicates that this k-packet's likelihood estimation for every VoIP application. The index of the maximum value in the vector is the identification result. We will set the value in map database and remove this flow in pending flows. We do not have a way to identify unknown VoIP applications for now, but we want to improve our system. So we will research the packets whose maximum likelihood is less than 0.9.

VIII. Performance evaluation

In this section, we first show the dataset used in our experiment. Secondly, we give the specific parameters used in the training of deep neural networks. Finally, we show the experimental results and give the evaluation of the system performance.

A. *Dataset*

We capture traffic and divide it into 2 dataset for these 10 VoIP applications. We use dataset1 to train identification models, and we use dataset2 to examine these models. The details of dataset1 used for training shown in Table I.

The forth column shows the num. of k-packet where k is 10, of course, we also process the dataset according to k is 2, 4, 6, 8 and so on. 80% of the k-packet are used for training when 20% used for validating. It decides the first dimension of the matrix that will be input into AlexNet model.

B. *Parameter Setting*

The optimization algorithm SGD used in training phase is not adaptive for learning rate, so the setting of learning rate has a direct impact on the experimental results. We learn from the general experience of deep learning, and we adjust it according

TABLE I
THE DETAILS OF DATASET1.

VoIP	Platform	Size(MB)	Num.(k=10)
Skype	Windows, Linux	3908.8	669984
UUCall	Windows	2709.4	691566
KCCall	Windows	3128.8	808224
ALTCall	Windows	2795.2	692002
Jumblo	Windows, Linux	3704.6	871468
Zoiper	Windows, Linux	4418.1	677114
Xlite	Windows, Linux	5165.3	638288
Eyebeam	Windows	4524.7	616773
ExpressTalk	Windows	4633.1	602637
Bria	Windows	4476.0	598724

TABLE II
THE DETAILS OF ALEXNET PARAMETERS.

Model	Input_Shape	Num. of Parm	Size(MB)
k=2	$2 \times 256 \times 1$	21827434	87.3
k=4	$4 \times 256 \times 1$	21827434	87.3
k=6	$6 \times 256 \times 1$	38604650	154.5
k=8	$8 \times 256 \times 1$	38604650	154.5
k=10	$10 \times 256 \times 1$	55381866	221.6
k=20	$20 \times 256 \times 1$	88936298	355.8
k=40	$40 \times 256 \times 1$	172822378	691.3
k=100	$100 \times 256 \times 1$	424480618	1619.3

to our experimental results. The initial learning rate is set to 0.01, it decays after 2 epochs following the rules mentioned in section 6.3. Nesterov updates based on the gradient, we set momentum factor to 0.9. We train 20 epochs using the dataset for the above 8 models, we set batch size to 100. With these parameters, we get satisfactory results for every model. Table II shows the number and storage size of weights identification model.

C. *Experiment Results*

The dataset listed in Section VIII-A is used to train 8 models. With the parameter setting in Section VIII-B, we get 8 identification models. The training and validating accuracies are shown in Table III.

For the above 8 identification models, we use TPR (True Positive Rate) and FPR (False Positive Rate) to evaluate them. TPR reflects the probability of positive samples are identified correctly, and FPR reflects the probability of negative samples are identified as positive samples incorrectly.TPR can be

TABLE III
THE TRAINING AND VALIDATING ACCURACY OF 8 IDENTIFICATION MODELS.

Model	Loss	Train Acc.	Validate Acc.
k=2	0.015865	0.995560	0.996130
k=4	0.007670	0.997700	0.997860
k=6	0.004933	0.998520	0.999340
k=8	0.004393	0.998690	0.999130
k=10	0.001159	0.999650	0.999650
k=20	0.000546	0.999880	0.999880
k=40	0.002920	0.999120	0.999460
k=100	0.006689	0.997620	0.998730

TABLE IV
THE TIME CONSUMED TO IDENTIFY 100 FLOWS.

Model	k=2	k=4	k=6	k=8	k=10	k=20	k=40	k=100
Time(s)	27.55	20.12	29.09	30.53	48.00	73.62	141.33	309.51

computed by Equation 5.

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

And FPR can be computed by Equation 6.

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

In our experiment, the identification rate for dataset1 is close to 1. When we examine these 8 models with dataset2, there are some outliers. They can be seen in the first 4 charts shown on Figure 5, model-20 for Skype, model-4 for Xlite and so on. This is overfitting, the identification model that we trained is overfitting the dataset. In the future research, we first consider add the L1-norm and L2-norm into our model, we will merge dataset2 into dataset1 to enlarge dataset to avoid overfitting. Apart from this, these charts show good performance to most of VoIP applications.

Figure 6 shows the probability distribution of true result and false result of dataset1 and dataset2, which true result means a k-packet is identified correctly and false result means a k-packet is identified in correctly. It shows the maximum probabilities of true results are generally close to 1, and they are disperse from 0.2 to 0.98. We use the feature set we extracted by neural network to train 4 machine learning models, the accuracy shows in Figure 7. The previous research we mentioned in Section II, they use machine learning and feature set extracted by human to identify VoIP traffic. With our feature set extracted by deep learning, the accuracy is higher than 98%.

The identification efficiency of SVM is better than original AlexNet model, we deploy identification model trained by SVM into real network. We capture 100 flows for each VoIP application and identify them by these models. Time consumed by these models is shown on Table IV. From this table we can see that it costs 3 seconds when k is 100, it is tolerate for a long call which lasts for a few minutes. For fraud calls which always end quickly, identification time needs to be further reduced, we have to consider using the smaller k.

IX. CONCLUSION

In our work, we propose a generic approach to identify VoIP application in realtime. Then we deploy this approach into a real-time identification system, and the system shows good performance. Our approach can not identify unknown VoIP applications, we will go a step further. We only deploy this system in a small network to test its real-time performance, we will deploy our system into real network and apply Storm in our system.

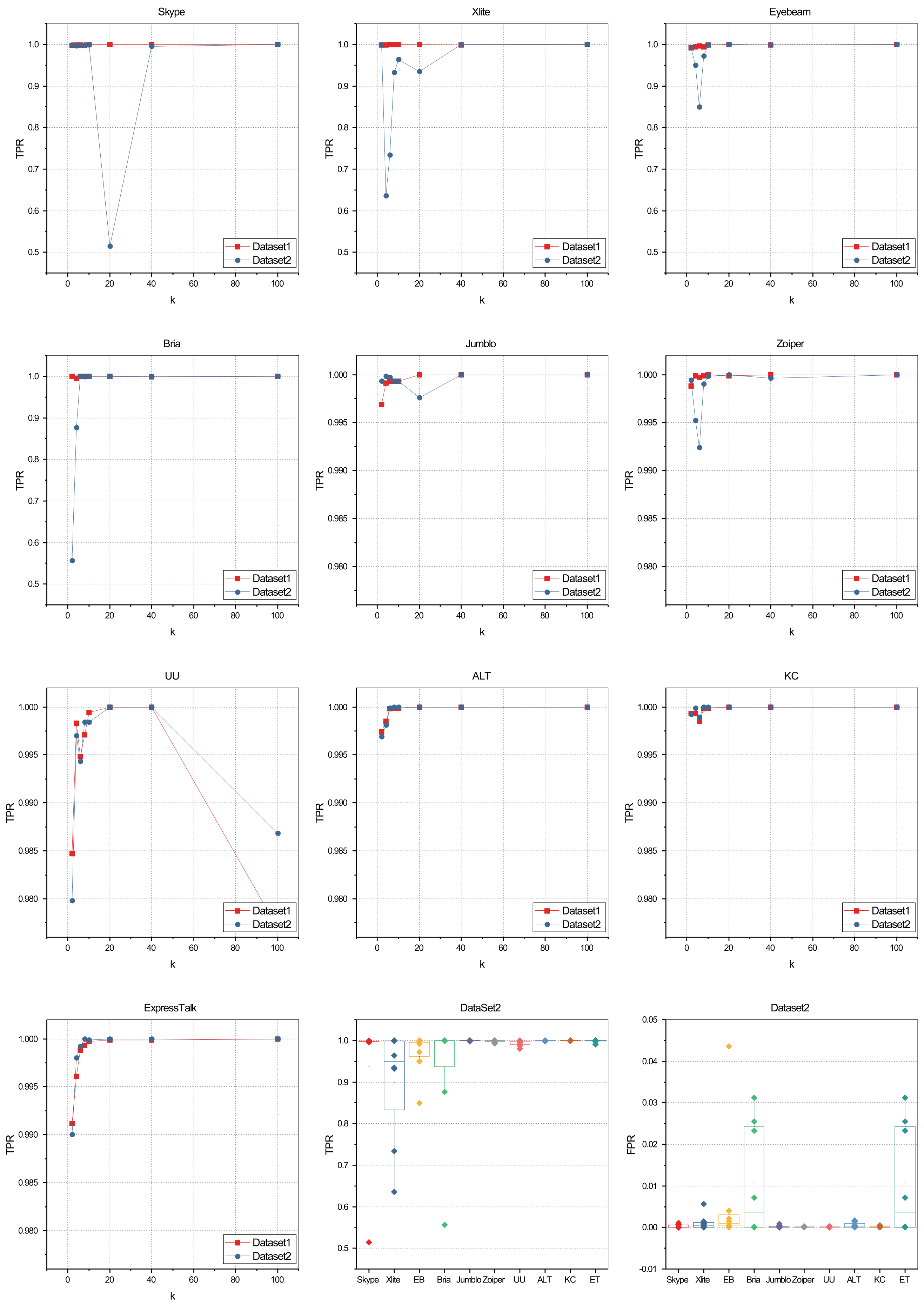

Fig. 5. The performance of the 8 models shown by TPR and FPR.

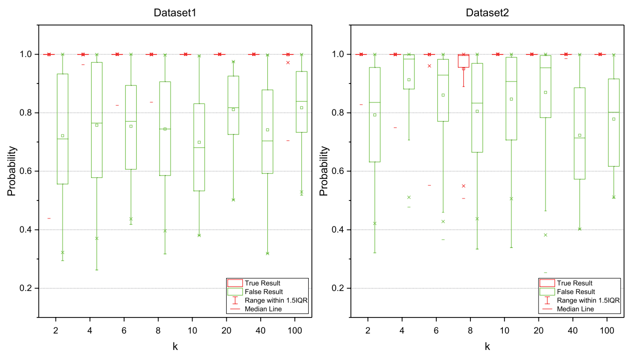

Fig. 6. The probability distribution of true result and false result.

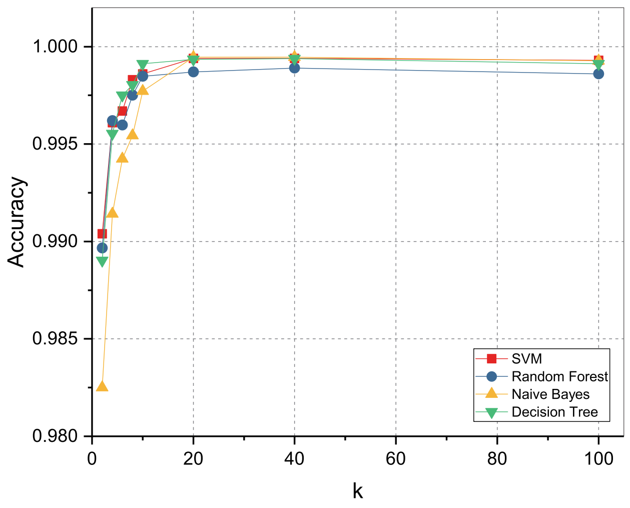

Fig. 7. The accuracy of 4 machine learning methods using extracted feature set.

References

[1] F. Idrees and U. Aslam, "A generic technique for voice over internet protocol (voip) traffic detection," 2008.

[2] T. Yildirim and P. J. Radcliffe, "Voip traffic classification in ipsec tunnels," in *International Conference on Electronics and Information Engineering*, 2010, pp. V1–151 – V1–157.

[3] J. V. Gomes, P. R. M. Inacio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Identification of peer-to-peer voip sessions using entropy and codec properties," *IEEE Transactions on Parallel & Distributed Systems*, vol. 24, no. 10, pp. 2004–2014, 2013.

[4] R. Alshammari and A. N. Zincir-Heywood, "Identification of voip encrypted traffic using a machine learning approach," *Journal of King Saud University - Computer and Information Sciences*, vol. 27, no. 1, pp. 77–92, 2015.

[5] Z. Wang, "The applications of deep learning on traffic identification," 2015.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors." *Readings in Cognitive Science*, vol. 323, no. 6088, pp. 399–421, 1988.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition*, 2016, pp. 770–778.