A real-time identification system for VoIP traffic in large-scale networks

Laboratory of Network and Information Security
College of Computer and Control Engineering, Nankai University
Tianjin, China
zhangjz@nankai.edu.cn, wangrongkang@mail.nankai.edu.cn

***Abstract*—Recently, VoIP applications win most of the users' favor with their high service quality and low price. However, owing to the flexible of VoIP services, it will cause incalculable damage if these services are used improperly. In order to make VoIP applications serve humans better, it is important to keep VoIP applications under supervision. The upgrading of VoIP technology makes the traditional identification method inefficient, it becomes more difficult to identify VoIP traffic. For the challenge of identifying VoIP traffic, the deep learning technology provides us a new solution. With supervised learning method, we can build feature set for identifing VoIP traffic. This way is not only able to find the most useful features, but also it can get rid of human effort in exploring features. In this paper, we adopt CLNN(Convolutional Neural Networks, Long Short-Term Memory) to extract features for accurate VoIP application identification. In addition, we design a real-time identification system to capture VoIP traffic in a large-scale network and identify their application types with the features we trained. The evaluation results verify that our system can identify VoIP traffic timely and accurately.**

***Index Terms*—VoIP, application identification, real-time, CNN, LSTM.**

I. Introduction

VoIP(Voice over Internet Protocol) is an internet technology for the delivery of voice communications and media sessions over IP networks, it provides an environment supports connection of all kinds of terminals via internet. Recently, with their price advantage, VoIP applications have been used more and more among the people. Most of VoIP applications can provide convenient services, such as anonymous calls, multi-line calls, call forwarding. They are so flexible that many unruly elements use VoIP services make fraud calls. The results of VoIP identification researches can help network administrators to enhance network even society security. For these reasons, VoIP identification has been an active research topic.

VoIP consists of the protocols that used in signaling and channel setup. H.323, SIP, MGCP and MEGACO are the most commonly used protocols for call signaling and controlling, RTP/RTCP is the most commonly used transmission protocol. Some researches [1]–[3] are devoted to analyze the behavior of the VoIP traffic generated in signaling phase, some [4], [5] are devoted to analyze the behavior of the voice traffic that RTP/RTCP carries. There are also some large manufacturers using their own VoIP protocol, such as the Skype protocol. Therefore, there are some researches [6]–[8] specifically identify Skype traffic. Unfortunately, encryption technologies such as SSL/TLS, WAP/WAP2 and SRTP/SRTCP is increasingly used to encrypt VoIP traffic. The utilization of encryption technologies has impact on both identification directions, identification task of VoIP traffic is getting harder and harder.

Encryption technologies are so mature that most VoIP applications already adopt them into signaling phase. Compared to signaling phase, SRTP/SRTCP require more computing resources than RTP/RTCP, considering calculation ability and transmission quality, most of the existing VoIP applications still adopt RTP/RTCP in voice transmission phase. As for encrypted voice traffic that few VoIP applications generate, we can extract more features than signaling traffic with deep learning method because of there are huge amounts of traffic generated in voice transmission phase. On the other hand, the development of MGCP and MEGACO protocols makes more VoIP applications adopt softswitch technology. It is unreliable to identify RTP/RTCP channel by analyze signaling traffic under NAT networks. For these reasons, we combine CNN and LSTM models which we called CLNN to extract features for flows generated during the voice transmission phase not the signaling phase. Considering identification real-time performance, we separate the VoIP entire flows into subflows to extract features. In our real-time system, after a subflow is identified, subsequent traffic belonging to this flow will be identified.

Our contributions are summarized as follow: 1) We designed CLNN model to extract features for voice flows, the CLNN model is able to extract features in both time and frequency domains, the extracted features are more reliable than extracted by humans. 2) We can identify a VoIP flow in acceptable time, we train subflow dataset instead of entire flow, it is a necessary condition for real-time identification. 3) We designed a real-time identification system to be deployed in large-scale networks, administrators can intercept VoIP traffic according to identification results.

The remainder of this paper is structured as follows. In section II, we describe the previously published related works. In section III, there are main methods used in our work. In section IV, we introduce the algorithms used in this paper. The performance evaluation is shown in section V. The conclusion is shown in section VI.

II. Related Work

Since early VoIP applications use the ports that IANA designated, some previous works identify VoIP traffic based on ports. With the development of VoIP applications, most of them use dynamic ports, the traditional identification method based on port numbers is not effective enough. To overcome these difficulties, some researches [1], [2] identify specify protocol with their regular characteristics, such as SIP, RTP. [9] uses DPI (Deep Packet Inspection) way to extract application-level signatures to detect VoIP applications. These kinds of works are too limited to identify private protocol like Skype protocol. Also, the above methods cannot identify VoIP applications adopt encryption technologies.

Statistical analysis method is another most frequently used method to detect VoIP traffic. Some characteristics such as packet-inter arrival time, packet size and rate of packet exchange are analyzed and used to detect VoIP traffic in [10]. Authors in [11] combine traffic flow statistic analysis with host behavior estimation. They set a parameter D to filter non-VoIP traffic out and calculate the R by EL (large inter-packet time) and ES (small inter-packet time) to identify VoIP traffic. There are still some latest researches [12], [13] adopt rule-based and statistical analysis-based solution to detect VoIP traffic. These ways require many thresholds to be set for corresponding rules to identify VoIP traffic, it's powerless for a wide variety of VoIP applications. With the development of machine learning, this method has gained its new vitality. In [14], authors propose the machine learning approach to identify encrypted VoIP traffic. They build flow-based feature set and employ 3 learning algorithms for training dataset, the results show a good performance to classify VoIP traffic. Usually this type of statistical analysis methods aim at detecting VoIP traffic, which means identify VoIP traffic in VoIP/non-VoIP level. With machine learning methods, it can achieve further classification in application level. The method proposed in [14] trains the classifier with entire flows, and it requires a lot of works to build feature set. So they hardly could be used for real-time identification.

There are several researches trying to identify VoIP traffic in real time. For detecting VoIP traffic quickly, the packets whose length in range of 60 to 150 bytes are treated as VoIP packets in [15]. This method only aims at improving VoIP service quality, it cannot be used for accurate VoIP identification. A method that identifies peer-to-peer VoIP sessions using entropy and codec properties is proposed in [4], they consider several applications and their corresponding codecs so that the classifier can identify a VoIP session with the specific codec it uses. Moreover, they focus on the relation between the different lengths of N packets and explore their level of heterogeneity using entropy. In order to guarantee the real-time of the classifier, they adopt a sliding window with a constant size of N packets to access the heterogeneity in real time. Same real-time identification method used in [16], authors extract the PSD (Packet Size Distribution) of the first few packets of the entire Bi-flow. Results show this method can identify Bi-flow quickly after its establishment. All such methods that real-time identify VoIP traffic are based on features of sub-flows, but the limited features is too powerless to identify VoIP traffic in a tiny time period.

Standing on the shoulders of prior researches and focusing on the difficulties we found, in this paper we try to extract features for huge amounts of data with deep learning technology. Based on the deep learning technology proposed in [17], we employ CLNN model to help us extract features automatically. These features are generic for non-encrypted and encrypted VoIP applications. The identification system we build can preliminarily detect VoIP traffic based on some rules, then the classifier can classify VoIP traffic into accurate application type.

III. Methodology

VoIP applications use RTP to transfer data, RTP typically runs over UDP. What we need to do is not only to identify RTP traffic from a large amount of UDP traffic, but also to identify VoIP traffic from RTP traffic. The ability to identify single VoIP packet is very low, so we identify VoIP traffic by identifying the flow it belongs to. Both offline training phase and online identification phase are based on this point, we introduce our methods for these 2 phases separately.

A. *Offline Training*

In offline training phase, we adopt supervised learning method to extract features, which are used in online identification phase. In order to verify VoIP flows carry the features supporting identification, we did some basic researches. Firstly, VoIP applications will select appropriate codecs based on network conditions or channel types, and the codecs supported by each VoIP application are different. The codecs we found in our dataset lists in Table I. The payload type in RTP header indicates the codec, this basic network feature can be used to identify VoIP traffic. Secondly, different codecs will occupy different bandwidth, in the meantime, different packing periods make VoIP applications to have different sending rates. Both PSD [16] and entropy [4] are related to codecs and packing times. In addition, ITD(Interarrival Time Distribution) and PTD(Packet Type Distribution) are different for each application. Except these features, there are many other features can be used for identifying.

TABLE I
CODECS USED BY VOIP APPLICATIONS.

Application	Codecs
Skype	dynamic (96-127), SILK, G.729, PCM A/U
UUCall	G.723, G.728
KCCall	PCM A/U, ILBC, GSM, G.729
Jumblo	PCM A/U, dynamic (118)
Zoiper	PCM A/U, GSM
Xlite	PCM A/U, iLBC, GSM, Speex
Eyebeam	PCM A/U
ExpressTalk	PCM A/U, Speex
Bria	PCM A/U, Speex, GSM

In order to design a model extracting the most useful features for VoIP flow identification, we reasearched the most popular CNN models such as LeNet-5 [18], AlexNet [19], VGG [20], GoogleNet [21] and ResNet [22]. CNN models have the ability to reduce frequency variations, so they are often used for digit recognition. By contrast, LSTM is good at temporal modeling and often used for speech recognition. LSTM as a development of RNN is proposed for solving the gradient problem in RNN model [23]. To extract the most useful features in both time and frequency domains, we design a deep learning model based on CNN and LSTM, we name it as CLNN model. Like many existing deep learning models, most of the CNN ideas come from AlexNet. The difference is we employ LSTM for improving temporal features. CNN and LSTM layers extract the features of frequency and time domain respectively before they are input into fully connected layers. Figure 1 shows how CLNN model works.

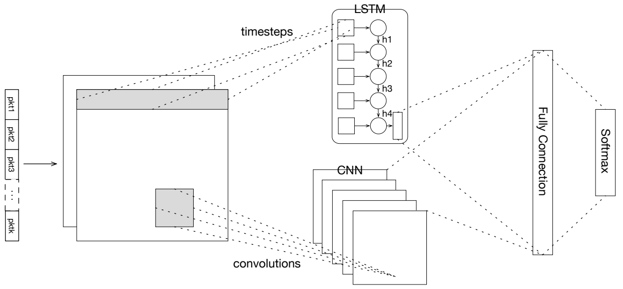

Fig. 1. Basic principles for extracting representations with CLNN model.

The way to extract features for entire VoIP flows cannot be used for real-time VoIP identification, we split entire VoIP flows into subflows. One subflow is consist of k continuous VoIP packets from the same channel, we use $subflow^k$ denoting one in the following sections. We design a sampling method that uses sliding window to select $subflow^k$ into training dataset, the sliding window can select its next step length randomly. This method not only promises the dataset is uniform but also reduces the training work.

B. Online Identification

The offline training work is the basis of online identification, the features we get in offline training are used for identifying VoIP flows. In online identification phase, we parse the $\{$srcIP, dstIP, srcPort, dstPort$\}$ in IP and UDP header, the quad-tuple is used to separate flows as a key. We generate a $subflow^k$ for each flow with their specific key, then we identify the $subflow^k$ with the online identification system. For each flow with specific key, the identification result is saved as the value of the key. The $\{$key, value$\}$ is saved into our database, we can identify the following traffic with their quad-tuple.

We design an online identification system for large-scale networks, there are 3 main components in this system. The basic architecture is shown in Figure 2. Capturer capture UDP packets and separate them into corresponding flows by IP address and UDP port. Filter follows several rules to filter out obvious non-VoIP flows. Classifier classifies flows into corresponding application type with extracted features.

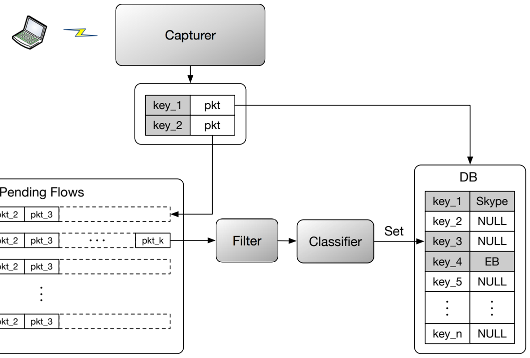

Fig. 2. Architecture of online real-time identification system.

We adopt Apache Storm as our stream processing computation engine, and Apache Kafka as backup and load balance system to interact with networks. Figure 3 shows the Storm topology of our system. There is a Kafka Spout as the Kafka consumer client to consume the traffic from the network. A Bolt as capturer, a Bolt to distribute pending flows to next Bolt, the subsequent Bolt may be a filter or a classifier, which is determined by our settings based on k. The last Bolt RedisWriter is in charge of writing the identification results to the Redis database, and in order to get more accurate identification result, the result in database will be replaced by the classifier that gets higher probability.

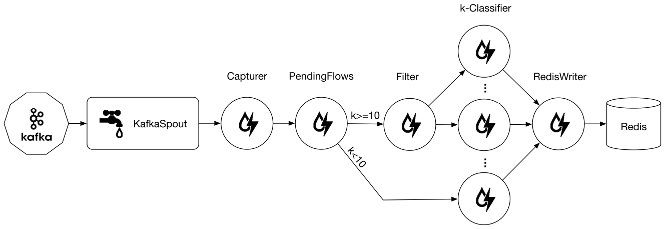

Fig. 3. Storm topology of online identification system.

IV. Algorithms

A. Data Preprocessing

We collect raw VoIP traffic and contruct $subflow^k$ dataset with our sampling method. We adopt different k as the size of sliding window to select $subflow^k$ into our dataset. When the window size is k, the next step is randomized in [1, k]. It ensures that our dataset isn't overly similar or vague. Figure 4 shows how we select 4 $subflow^k$ for $k = 10$.

The $subflow^k$ we select are processed to matrices, these matrices as training data are input into CLNN model. We have 5 steps to convert a $subflow^k$ to matrix. 1) Remove IP and UDP header; 2) Add 2 timestamps to each packet, one is the time difference from the first packet, the other one is the time difference from the previous packet; 3) Add 0 to denote forward direction of the traffic transmission, 255 to denote

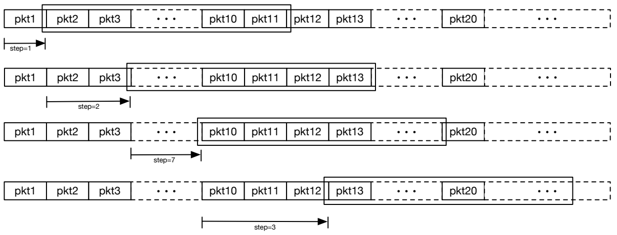

Fig. 4. How to process raw traffic to build $subflow^k$ dataset(k=10).

the backward direction; 4) Convert the $subflow^k$ to matrix on the basis of ASCII code; 4) Normalize the matrix, the timestamps and direction types what we add are normalized too. The length of VoIP packets is generally small around 50∼210, so we set column of the matrix to 256. The row of the matrix is k, indicating a $subflow^k$ includes k packets.

B. CLNN model

Based on section IV-A, the input of CLNN model is matrices with size of $k \times l$, where k is number of packets used for $subflow^k$, l is the column of matrix, we use 256. We empoly 2 LSTM layers, where each layer has 512 hidden cells. And we employ 5 CNN layers in parallel, the parameters of each CNN layer is similar with AlexNet. We adjust the filter kernel to 5×5 in the first layer, and we change the padding type to same in each layer. We only use 1 max pooling after the first layer. After the 5$^{\text{th}}$ CNN layer, there are 1 flatten layer to reshape features into one-dimension and 1 fully connection layer to extract features preliminary. The features extracted by LSTM and CNN parts are concatenated and input into fully connection layers, the fully connection layer has 2048 hidden units. The concatenated features are input into fully connection layer with 2048 hidden units. The last layer as the output layer is activated using Softmax function, the sample will be classified into 11 classes.

C. Learning

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,

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

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 at time t.

In our work, learning rate decays after each epoch, decay rule is shown as:

$$\eta_i = \eta_{i-1} \times \frac{1}{1 + \rho \times i} \tag{2}$$

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

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

$$CCE(V_i) = -\sum_{i=1}^{n} P(V_i) \times \log(\hat{P}(V_i)) \tag{3}$$

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 $subflow^k$ belongs.$P(V_i) = 0$,where $i \neq j$.

We use Softmax function as the activation function in the last layer. There are 11 types of traffic in our experiment, we calculate the posterior probability for unknown flow to identify its application type. Softmax function shows as

$$\hat{P}(V_i) = softmax(V_i) = \frac{e^{V_i}}{\sum\limits_{j=1}^{n} e^{V_j}} \tag{4}$$

where $(V_1, V_2, \cdot\cdot\cdot, V_i, \cdot\cdot\cdot, V_n)$ denotes the matrix to be inputted into Softmax in the 8th layer; $(\hat{P}(V_1), \hat{P}(V_2), \cdot\cdot\cdot, \hat{P}(V_i), \cdot\cdot\cdot, \hat{P}(V_n))$ is the output of the 8th layer, it is posterior probability distribution of $subflow^k$, it is a vector with size of $1 \times n$; n is 11 in our experiment, it denotes the number of training traffic types.

D. Components in Identification System

The first component capturer maintain a pending flows queue and map packet into its matching pending flow using its IP address and UDP port. Figure 5 shows the details to separate flows by IP address and UDP port. If a packet cannot find the corresponding pending flow, it creates a new pending flow and adopt IP address and UDP port as its key. If a packet is appended into pending flow, and the number of packet reaches k, this $subflow^k$ will be sent to next component filter. The capturer guarantees that all captured packets are UDP packets, and the capturer can remove dead flows from pending flows queue. Flows whose number of packets doesn't reach k within limited time will be considered as dead flows.

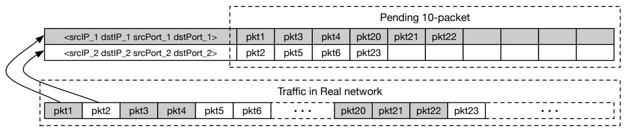

Fig. 5. To separate flows by IP address and UDP port in real network.

The second component is a rule-based filter, it receives $subflow^k$ sent by capturer. Because non-VoIP flows are no value to us, it will be filtered out to a certain extent. The remaining flows will be sent to next component to do further identification. We select benchmark values from our dataset and set several rules based on these benchmark values. Related algorithm and parameters are shown in Algorithm 1 and Table II.

TABLE II
BENCHMARK VALUE AND THRESHOLD USED IN ALGORITHM 1.

	Symbol	Description
1.	min_mpl	Minimum of mean packet length in datatset
2.	max_mpl	Maximum of mean packet length in datatset
3.	min_pl	Minimum of packet length in datatset
4.	max_pl	Maximum of packet length in datatset
5.	min_miat	Minimum of mean inter-arrival time in datatset
6.	max_miat	Maximum of mean inter-arrival time in datatset
7.	min_iat	Minimum of inter-arrival time in datatset
8.	max_iat	Maximum of inter-arrival time in datatset
9.	λ_1, λ_2	2 thresholds for upper and lower limits
10.	ϱ	Proportional threshold for number of packets

Algorithm 1 Algorithm to identify VoIP/non-VoIP Flows

```
Input: Flow: $subflow^k$
Output: VoIP/non-VoIP
if $\lambda_1$min_mpl < mean packet length of current $subflow^k$ < $\lambda_2$max_mpl and $\lambda_1$min_miat < mean inter-arrival time of current $subflow^k$ < $\lambda_2$max_miat then
    for packet in $subflow^k$ do
        if (packet length < $\lambda_1$min_pl ) then
            $ct\_pl\_l \leftarrow ct\_pl\_l + 1$
        end if
        if (packet length > $\lambda_2$max_pl ) then
            $ct\_pl\_u \leftarrow ct\_pl\_u + 1$
        end if
        if (packet inter-arrival time < $\lambda_1$min_iat ) then
            $ct\_iat\_l \leftarrow ct\_iat\_l + 1$
        end if
        if (packet inter-arrival time > $\lambda_2$max_iat ) then
            $ct\_iat\_u \leftarrow ct\_iat\_u + 1$
        end if
    end for
    if ct_pl_l < $\varrho$k and ct_pl_u < $\varrho$k and ct_iat_l < $\varrho$k and ct_iat_u < $\varrho$k then
        return $VoIP$
    else
        return $non-VoIP$
    end if
else
    return $non-VoIP$
end if
```

The $subflow^k$ that pass through filter successfully will be input into the third component classifier, the features used by the classifier are extracted in offline training phase. 4 kinds of classifiers are trained by SVM, Random Forest, Naive Bayes and Decision Tree, we select SVM classifier which have the best performance into our real-time system. The classifier takes k-dimensional matrix as input and 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 a $subflow^k$'s likelihood estimation for every VoIP application. The index of the maximum value in the vector is the identification result. The classifier can set the identification result in database, subsequent packets will be identified by querying the database. In the process of identification, one flow will pass through multiple k-classifiers when it grows up to different k stages, these classifiers will give their own identification results and corresponding possibilities.

V. PERFORMANCE EVALUATION

In this section, we introduce the dataset used in our experiment firstly. Secondly, we give the specific parameters used in the CLNN model. Finally, we show the experimental results and give the evaluation of the system performance.

A. Dataset

Collecting VoIP traffic 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 phones ready to be called. We try our best to make calls from various places in the country to improve the credibility of our experiment. Besides, we collect the traffic from various video and game software as non-VoIP traffic. We divide the captured traffic into two equal parts, the first part is preprocessing into subflow dataset when the second part is used for replaying by tcpreplay software to examine online identification system.

Obtaining pure dataset and labeling dataset accurately are the basis of training good model. We adopt some traffic capture tools which are process-based. They can help us get pure raw traffic for each application, and we can label our dataset using the name of VoIP applications. 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. 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 analyze the ports VoIP application uses, then we use command like "tcpdump -i eth_name trans_protocol port port_id" to capture traffic of specific ports. Because of process-based and port-based capture methods, it is easy to get pure raw traffic and label subflows with their corresponding process name.

We construct $subflow^k$ dataset with 8 different k, including 2, 5, 10, 20, 30, 40, 50 and 100. It is used for training 8 CLNN models. The details of training dataset is shown in Table III. The forth column shows the number of $subflow^k$ where k is 10, definitely, we also process the dataset according to k is 2, 5, 10, 20 and so on. It decides the first dimension of the matrix that will be input into CLNN model. 80% of the $subflow^k$ are used for training when 20% used for validating in training pahse.

B. Training Setting

The optimization algorithm SGD used in training phase is a learning rate non-adaptive algorithm, the setting of learning

TABLE III
THE DETAILS OF $subflow^{10}$ DATASET.

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
non-VoIP	Windows	11324.5	973522

TABLE IV
THE DETAILS OF CLNN PARAMETERS.

Model	Input_Shape	Num. of Parm	Size(MB)
k=2	$2 \times 256 \times 1$	17,818,697	87
k=5	$5 \times 256 \times 1$	17,820,823	87
k=10	$10 \times 256 \times 1$	34,600,197	154
k=20	$20 \times 256 \times 1$	34,602,387	154
k=30	$30 \times 256 \times 1$	51,381,825	221.6
k=40	$40 \times 256 \times 1$	84,947,847	355.8
k=50	$50 \times 256 \times 1$	168,859,507	691.3
k=100	$100 \times 256 \times 1$	420,613,687	1619.3

rate has a direct impact on the experimental results. We learn from the general experience of deep learning, and we adjust it according 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 the dataset 20 epochs with batch size 50, 100 and 200. Table IV shows the number and storage size of weights identification model.

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.

C. Experiment Results

The dataset listed in Section V-A is used to train, the numbers of subflows used to train are same for 8 different k. With the parameters setting in Section V-B, we get 8 CLNN models. The training and validating accuracies are shown in Table V. As the table shows, identification accuracy is correlated with batch size and packets number. Smaller batch size is able to get higher accuracy, but it takes more time in training phase. Identifying subflows with more packets will get higher accuracy, but a higher k means also reduce the real-time performance of identification system. We need to balance the training time, real-time performance and identification accuracy in different scenarios.

For the above 8 CLNN models, we use TPR (True Positive Rate) to evaluate them. TPR reflects the probability of positive samples are identified correctly.

Figure 6 shows the performace of the 8 CLNN models for each VoIP application. VoIP traffic with obvious identification features can be accurately identified even k value is small, the accuracies of Skype, Xlite and Jumblo can reach above 90% when k is 2. VoIP applications with very similar statistical features need more packets to support identification. For example, both Eyebeam and ExpressTalk adopt ITU-T G.711 PCMU as their codec, but these 2 VoIP applications adopt different packing times. So identifying them requires more temporal domain features, which means a higher k. The last chart shows the probability distribution of the true samples, which true samples means a $subflow^k$ is identified correctly. The connection line is connecting mean, it reflects a higher k can get a more credible identification results.

The first chart in figure 7 shows the probability distribution of true samples and false samples(batch size=100). The CLNN models don't have the ability to identify unknown VoIP samples, so we identified 1000 known VoIP and 1000 unknow VoIP samples. The probability distribution of known VoIP samples and unknown VoIP samples shows in the second chart. We can see the probabilities of the true samples or known VoIP samples are generally greater than false samples and unknown VoIP samples, it's a condition to judge credibility of identification results.

We use the feature set we extracted by CLNN model to train classifiers with 4 machine learning models, the accuracy of them are shown in Figure 8. The previous research we mentioned in Section II, they use machine learning methods and feature set extracted by human to identify VoIP traffic. With our feature set extracted by CLNN model, the accuracy is up to 98% or even higher when k is higher than 10.

SVM has the best identification performance in these 4 machine learning methods. Although the accuracy of SVM is slightly lower than the CLNN model, but its real-time performance is better than original CLNN model. So we deploy classifiers trained by SVM into real network. We count the time consumed for identifying 100 VoIP flows with our system. Time consumed is shown on Table VI. 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 minutes. For that call which end quickly, make identification time shorter is more valuable, classifiers with smaller k will obtain identification results faster. In general, the real-time performace is enough for VoIP long connections.

VI. CONCLUSION

In our work, we propose a generic approach to real-time identify traffic of specific VoIP applications. We train classifiers with feature set extracted by CLNN model and deploy these classifiers into our real-time identification system. The real-time system is deployed with stream processing computation framework Apache Storm and it shows good performance. For now, the VoIP applications mentioned in this paper only cover the applications that can directly dial cell phone, some other applications using the VoIP protocol may be judged wrong or judged as non-VoIP applications. Our next goal is to enrich the types of VoIP applications. Similarly, the types of non-VoIP traffic need to be as rich as possible too.

TABLE V
THE TRAINING AND VALIDATING ACCURACY OF 8 CLNN MODELS.

Model	**Batch Size=50**			**Batch Size=100**			**Batch Size=200**		
	Loss	Training Acc.	Validating Acc.	Loss	Training Acc.	Validating Acc.	Loss	Training Acc.	Validating Acc.
k=2	0.522122	0.823860	0.834630	0.795628	0.658820	0.672410	0.901531	0.605440	0.635330
k=5	0.063926	0.989410	0.997860	0.320275	0.874420	0.893790	0.781780	0.627210	0.647860
k=10	0.016822	0.996520	0.999340	0.091234	0.987550	0.999340	0.493562	0.924250	0.939340
k=20	0.008969	0.998390	0.999130	0.034948	0.994830	0.999130	0.411541	0.949210	0.949130
k=30	0.018781	0.998150	0.999650	0.018984	0.997710	0.999650	0.246316	0.983630	0.999650
k=40	0.004145	0.998620	0.999880	0.007481	0.998520	0.999880	0.028801	0.997570	0.999880
k=50	0.001125	0.999730	0.999460	0.014981	0.996530	0.999460	0.144059	0.992870	0.999460
k=100	0.000329	1.000000	1.000000	0.001795	0.999620	0.999730	0.059798	0.978120	0.998730

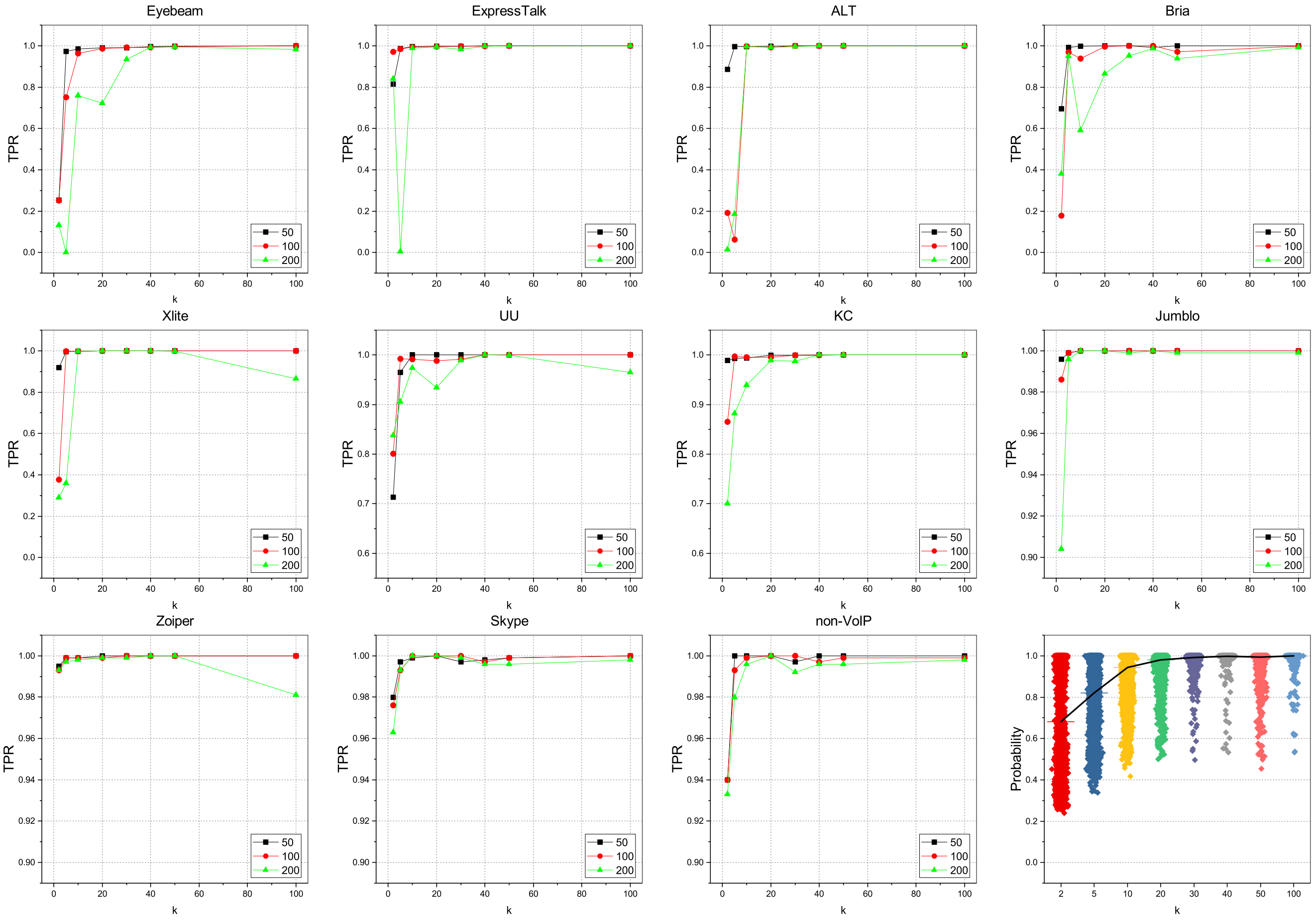

Fig. 6. The performance of the 8 CLNN models shown by TPR.

TABLE VI
THE TIME CONSUMED TO IDENTIFY 100 FLOWS.

Model	**k=2**	**k=5**	**k=10**	**k=20**	**k=30**	**k=40**	**k=50**	**k=100**
Time(s)	27.55	20.12	29.09	30.53	48.00	73.62	141.33	309.51

Another weak point is that training is too time consuming, if network administrators want to add new VoIP applications, the feature set need to be trained again. Given that the main goal of our mission is to intercept specific VoIP traffic in real time, the current system has achieved new breakthroughs in identification accuracy.

REFERENCES

[1] J. Curtis, J. Cleary, A. McGregor, and M. Pearson, “Measurement of voice over ip traffic,” in *Passive and Active Measurement conference*, 2000.

[2] M. Chen, G.-X. Zhang, and J.-P. Bi, “Research of sip-based voip traffic identification methodology,” *Appl. Res. Comp*, vol. 24, 2007.

[3] J. Lee, K. Cho, C. Lee, and S. Kim, “Voip-aware network attack detection based on statistics and behavior of sip traffic,” *Peer-to-Peer Networking and Applications*, vol. 8, no. 5, pp. 872–880, 2015.

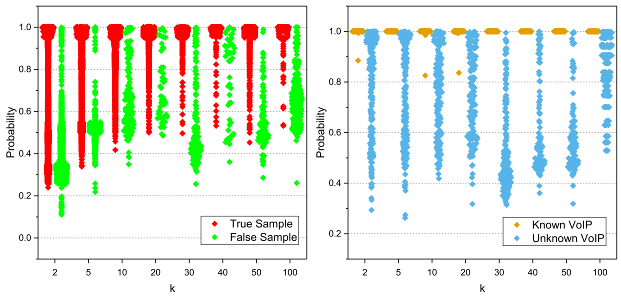

Fig. 7. The comparison of probability distribution for different samples.

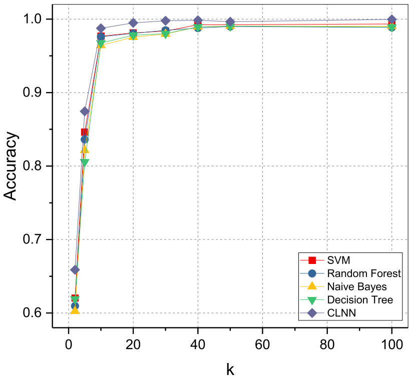

Fig. 8. The accuracy of 4 machine learning methods using extracted feature set.

[4] 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.

[5] T. Qin, L. Wang, Z. Liu, and X. Guan, “Robust application identification methods for p2p and voip traffic classification in backbone networks,” *Knowledge-Based Systems*, vol. 82, pp. 152–162, 2015.

[6] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, “Detailed analysis of skype traffic,” *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 117–127, 2009.

[7] D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe, “Skype-hunter: A real-time system for the detection and classification of skype traffic,” *International Journal of Communication Systems*, vol. 25, no. 3, pp. 386–403, 2012.

[8] S.-H. Lee, Y.-H. Goo, J.-T. Park, S.-H. Ji, and M.-S. Kim, “Sky-scope: Skype application traffic identification system,” in *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*. IEEE, 2017, pp. 259–262.

[9] S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in-network identification of p2p traffic using application signatures,” in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 512–521.

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

[11] B. Li, Z. Jin, and M. Ma, “Voip traffic identification based on host and flow behavior analysis,” in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*. IEEE, 2010, pp. 1–4.

[12] M. Rathore, A. Paul, A. Ahmad, M. Imran, and M. Guizani, “High-speed network traffic analysis: detecting voip calls in secure big data streaming,” in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 595–598.

[13] N. A. Saqib, Y. SHAKEEL, M. A. Khan, H. MEHMOOD, and M. Zia, “An effective empirical approach to voip traffic classification,” *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, no. 2, pp. 888–900, 2017.

[14] 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.

[15] 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.

[16] T. Qin, L. Wang, Z. Liu, and X. Guan, “Robust application identification methods for p2p and voip traffic classification in backbone networks,” *Knowledge-Based Systems*, vol. 82, pp. 152–162, 2015.

[17] Z. Wang, “The applications of deep learning on traffic identification,” 2015.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.