Mobile Application Identification Over HTTPS Traffic Based on Multi-view Features

Mao Tian*†, Peng Chang*†, Yafei Sang*†, Yongzheng Zhang*†, Shuhao Li*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
Email: {tianmao, changpeng, sangyafei, zhangyongzheng, lishuhao}@iie.ac.cn

***Abstract*—The expanding volume of HTTPS traffic (both legitimate and malicious) creates even more challenges for mobile network security and management. In this work, we propose AIBMF(Application Identification Based on Multi-view Features), a fine-grained approach to classify HTTPS traffic by their application type. The key idea of AIBMF is to combine three kinds of features—payload convolution features, packet size sequence and packet content type sequence. Based on these different view features, a deep learning model (using CNN, embedding and RNN) is constructed for HTTPS traffic identification task. To evaluate the effectiveness of AIBMF, we run a comprehensive set of experiments on a real-world dataset (about 100,000+ flows), which shows that our approach achieves 96.06% accuracy and outperforms the state-of-the-art method (3.6% on F1 score).**

***Index Terms*—HTTPS, mobile application, CNN, embedding, RNN.**

I. Introduction

HTTPS(Hyper Text Transfer Protocol Secure) usage continues to grow explosively for network security considerations. For example, Google is investing and working to make sure that its sites and services provide modern HTTPS by default adhere to the belief that security is a top priority at Google. Its goal is to achieve 100% encryption across their products and services [1]. HTTPS traffic classification has been a task of crucial importance for many purposes, such as QoS guarantees, network anomaly detection, privacy protection and so on. Recently, the percentage of encrypted traffic has risen rapidly. This change poses a challenge to currently used methods for traffic measurement, for which the identification and analysis of network traffic become more difficult [2]. The existing technologies for HTTPS traffic identification generally employ machine learning methods based on statistical features [3] [4], the main drawback of this kind of approaches is the complexity of feature design and selection, which will lead to poor general applicability when the network environment changes. There are also some other approaches leaving alone statistical features. [5] made use of TLS-embedded message type as state and established hidden Markov models on packet level. [6] presented a method mainly based on SNI(Server Name Indication). However, SNI information leaked out from HTTPS client Handshake can be easily faked and may be dismissed in some clients [7]. Additionally most existing methods focus on coarse-gained classification for protocols or services identification which is far less as for the fined-gained classification of network traffic required.

Given the lack of current research on encrypted traffic identification, we propose a method—AIBMF for associating HTTPS traffic with a specific application. It belongs to fine-grained traffic identification. Our method combines the payload information, packet size sequence and content type sequence together to complete the identification task. Payload is the part of transmitted data that is the actual intended message. When data is sent over the Internet, each unit transmitted includes both header information and the actual data being sent. The header identifies the source and destination of the packet, while the actual data is referred as payload. Packet size changes during the communication process as each packet contains different content. Content type is time-varying parameter of each packet, [8] defines the meaning of content type of a packet, for example, integer *20*, *21*, *22*, *23*, *24* is referred to *change_cipher_spec*, *alert*, *handshake*, *application_data*, *heartbeat* respectively. In AIBMF, content type can be regarded as a state of the current packet in a flow. We use 1D-CNN(one-dimensional convolutional neural network) to process payload and packet size sequence and use embedding and RNN(recurrent neural network) to process content type sequence to obtain abstract features. Our model combines these three types of features to complete the application identification task and the experiment results shows that our method achieves the best performance which outperforms the baseline [9] with more than 2.0% in $Precision$, $Recall$, and F_1 $Score$.

The main contributions of this paper are summarized as follows:

- This paper propose a application identification method over HTTPS traffic called AIBMF. We first combine the payload, packet size sequence and content type sequence together and use deep learning methods to extract valuable information from multiple views.
- This paper proposes a set of HTTPS traffic feature extraction methods. We study the effect of payload length of each packet and the packet counts of each flow and extract abstract features by CNN, embedding and RNN. The feature extraction method is very simple without conflict definitions, so the human effort is very small and these features are also universal in different network environment.
- We establish a HTTPS traffic dataset for 20 popular apps. We use our dataset to train different methods for identifi-

cation task. The evaluation result shows that our proposed method—AIBMF can achieve a higher performance.

The rest of this paper is organized as follows. Related works are presented in section II and the AIBMF framework is briefed in section III. Experiments and evaluations are shown in Section IV. Section V puts forward discussion and conclusion is made in the last section.

II. Related work

There have been a rich literature dealing with network traffic classification. For HTTP traffic, the idea of using the statistical properties of network traffic to classify flows is not new. Moore et al. [10] have studied 248 statistical features in TCP(Transmission Control Protocol) flow. Based on these features, the supervised learning method, Naive Bayes, is used to accurately classify the traffic. Wright et al. [11] proposed a feature extraction method based on size and time. This method is completely independent of the content. Fahad et al. [12] proposed a method for selecting traffic characteristics, the main purpose of which is to obtain the most important traffic characteristics. The approach shown by Jonathan et al. [3] focus on the statistical analysis of encrypted network traffic for identifying user's operating system, browser and application. They present new features that exploit browsers' bursty behavior and SSL(Secure Sockets Layer) behavior. These methods based on statistical features rely on complex feature construction and extraction.

Meng Shen [5] et al. proposed a traffic classification method based on Second-Order Markov chain and application attribute bitgrams. It mainly utilizes the message type of HTTPS which is a protocol designed feature in the communication process as the state information, and extends the fingerprint of the application. And it verifies that the message type feature of the packet is useful for traffic identification. Yu Zhang et al. [13] proposed a method for identifying automatic mobile applications based on HTTP traffic. They used convolutional neural network to automatically extract the features and established a binary classification model for each application. However, the application market has a large number of applications, the time complexity of model training and identification for all applications will be very high. Wei Wang [9] et al. proposed an end-to-end traffic classification method that extracts the first 784 bytes of a flow or session neglecting any other information, and then uses one-dimensional convolution for processing, our work is closely related to this study and we use it as baseline comparing with our method. However our statistics show that the length of a packet may exceed 784 bytes in a flow, hence this method may only extract feature from one packet and will neglect the other packets of a flow, this work failed to achieve a high performance for detailed encrypted traffic classification in our experiment.

III. AIBMF Framework

Figure 1 shows the overview of our proposed detailed HTTPS traffic identification method, consisting of preprocessing phase, feature extraction phase, and training & identification phase.

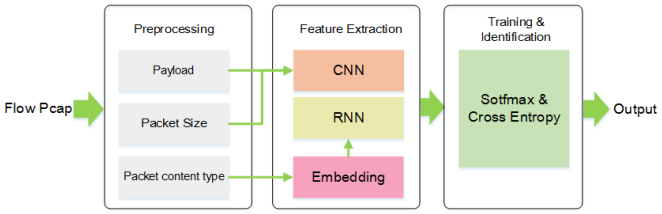

Fig. 1. Overview of Application Identification Process

In the first phase, we extract payload , packet size sequence and content type sequence of a flow from pcap files. In the second phase, some deep learning methods are applied to extract features from the preprocessed data. In the third phase, we train our model and complete identification tasks.

A. Data Preprocessing

We identify applications based on HTTPS flows. The 5-tuple information—$\langle Src\ IP,\ Dst\ IP,\ IP\ protocol,\ Src\ Port,\ Dst\ Port\rangle$ is used to identify a flow. We use the Splitcap tool [14] to segment the captured pcap files by flow, and each flow is saved as a new pcap file. We extract payload, packet size sequence and content type sequence from each pcap files and save these data into TFRecord which is a binary file format for storage of data.

1) Payload: In the preprocessing stage, open source tool Scapy [15] is used to read pcap files, and each pcap file is loaded as an ASCII string. We convert each ASCII code to an integer of 0—255. We conducted a series of comparative experiments shown in section IV to decide how many packets in a flow and how long of each packet to use. Finally, the first 16 packets and first 64 bytes payload of each packet are taken, and the packet less than 64 bytes are supplemented with integer -1. The payload information is converted into a $16 \times 64 = 1024$ long integer vector which is also padded by integer -1 when shorter than 1024.

2) Packet size sequence: Packet size changes during communication for each packet contains different content, therefore, packet size sequence is a kind of serial feature. Packet size is simple statistical feature which need little effort to extract. In our method, the packet size of the first 16 packets are recorded and stored in a 16-long integer vector(padded with -1).

3) Content type sequence: We use content type [8] which is protocol designed and time-varying during the communication as state information in our method. It ranges ranges from 0 to 255. The packet content type of the first 16 packets are recorded and stored in a 16-long integer vector(padded with -1).

B. Features Extraction

This paper proposes a model for traffic classification, as shown in Figure 2, which combines payload, content type sequence and packet size sequence features using 1D-CNN on the payload and packet size sequence, embedding and RNN on the content type sequence.

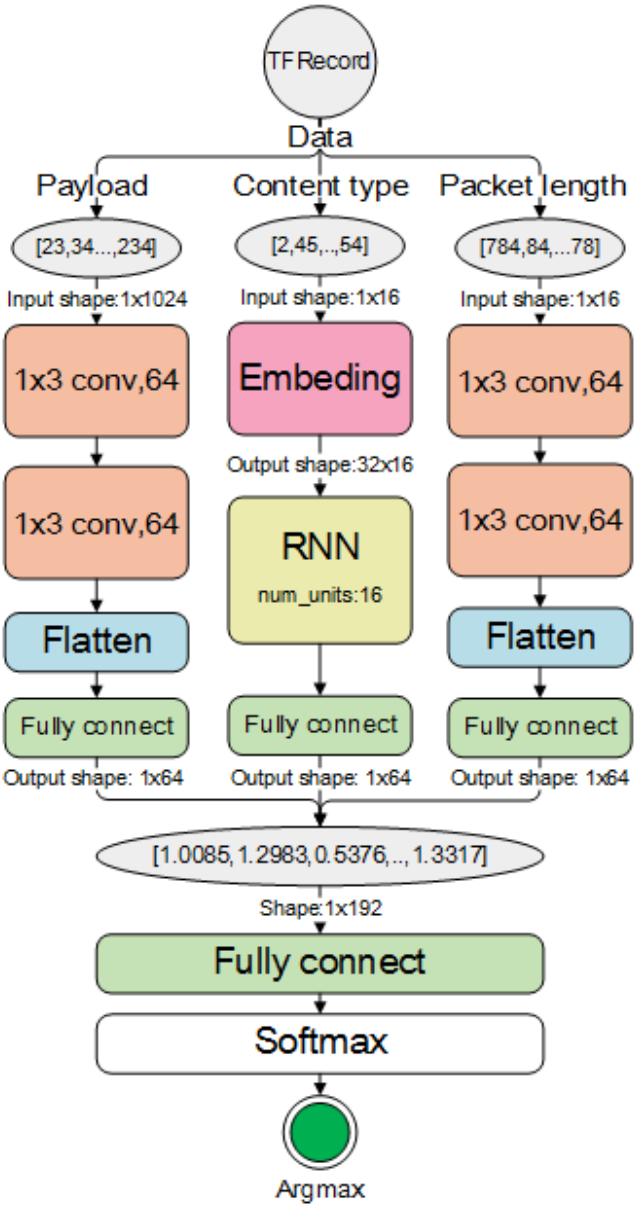

Fig. 2. AIBMF Architecture

1) Features from payload: Yoon Kim [16] used the 1D-CNN to learn features for sentence classification. We use a 1D-CNN to process the payload and packet size information of a flow inspired by this work. We use a two-layer convolutional neural network. The number of convolution kernels used in each layer is 64, and the size of the convolution kernel is 1×3. A typical layer in a convolutional network contains three levels [17]. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, we choose ReLU(rectified linear) activation function in this paper. In the third stage, we use a pooling function to modify the output of the layer further.

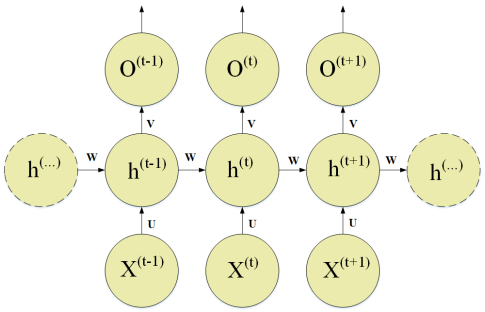

Fig. 3. A Recurrent Neural Network(RNN). Three time-steps are shown.

The pooling function replaces the output of the network at this location using the statistical characteristics of the adjacent output at a certain location. The maximum pooling function is used in this paper. We let $x_{i:i+j}$ refer to the concatenation of ASCII code of payload $x_i, x_{i+1}, ..., x_{i+j}$. The convolution kernel $\boldsymbol{w}$ is defined in the real space $\boldsymbol{w} \in \mathbb{R}^{hk}$, h refers to the length of the convolution kernel, and k refers to the dimension of the original data. Here, $h = 3, k = 1$, each feature c_i is calculated from a sequence window $x_{i:i+h-1}$:

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

Here $b \in \mathbb{R}$ is a bias term and f is a nonlinear function. The convolution kernel is applied to all windows $\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$ of the packet payload and packet size sequence in the flow to produce a feature map.

$$\boldsymbol{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{2}$$

Here $\boldsymbol{c} \in \mathbb{R}^{n-h+1}$, $n = 1024$ is the input length. Then we apply max pooling on the result of the convolution calculation and take the maximum $\hat{c} = max(\boldsymbol{c})$ as the result of the convolution calculation. After two-layers convolution, payload is transformed into a 64-long vector.

2) Features from packet size sequence: Packet size sequence is processed by 1D-CNN the same as payload, the differences are that the input vector is packet size sequence and the input length is 16 here. After two-layers convolution, packet size sequence is transformed into a 64-long vector.

3) Features from content type sequence: In the process of data transmission using HTTPS, the content type of each packet changes with time. The content type is originally a discrete variable represented by a 257-dimensional one-hot vector which is very sparse, so we introduce the embedding operation which maps the sparse representation to a 32-dimensional real vector which incorporates contextual information about the content type. Embedding mathematically represents a mapping: $f : X \rightarrow Y$, which satisfies two properties: injective and structure-preserving. The Embedding layer needs to learn a transform matrix whose shape is $(vocab\_size, embedding\_dim)$.

RNN [18] is a kind of neural network for processing sequence data. Figure 3 introduces the RNN architecture. For each time step from $t = 1$ to $t = 16$, we apply the following

update equations, the input is $x^{(t)}$ which is the content type embedding vector, the hidden layer activations are $h(t)$, the outputs are $o^{(t)}$.

$$\alpha^t = b + Wh^{(t-1)} + Ux^{(t)} \tag{3a}$$

$$h^{(t)} = tanh(\alpha^{(t)}) \tag{3b}$$

$$o^{(t)} = c + Vh^{(t)} \tag{3c}$$

where the parameters are the bias vectors b and c along with the weight matrices U, V and W , respectively for input-to-hidden, hidden-to-output and hidden-to-hidden connections. After Embedding and RNN, content type is transformed into a 64-long vector.

C. Training and Identification

Three different abstract features are concatenated together to form a 192-long real number vector which is then input to the last fully connected layer. The output of the last fully connected layer,i.e.,x, will be used as the input of the softmax regression to identify the flow. A flow can be identified into K classes, where $M = K$ denotes all type of apps. Three different abstract features jointly optimize the same category target, jointly calculate the multi-class cross entropy loss, and update the parameters of the three neural networks through the back propagation algorithm [18]. Mathematically, the probability that an output prediction Y is app i, is determined by:

$$p(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{w_i^T Z + b_i}}{\sum_j e^{w_j^T Z + b_j}} \tag{4}$$

where W is a weight matrix between the last fully connected layer and the Softmax layer, and the b is the bais vector. Then the model's prediction y_{pd} is the class whose probability is maximal:

$$y_{pd} = argmax(p(Y = i|x, W, b)), \forall i \in \{1, 2, 3, ..., K\} \tag{5}$$

Our neural network selects the cross entropy [19] as the loss function. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. It is defined as:

$$H(y, p) = -\sum_{i}^{M} y_i log(p_i) \tag{6}$$

where M is the number of apps, p_i is the probability calculated by softmax regression.

Over-fitting is a common problem in deep learning. Generally, the training is excessive, and the neural network is too complicated to have a good performance. In AIBMF, we choose to deploy Dropout [20] to reduce the over-fitting. Each training discards 20% of the neuron information, and all neurons are used in the test. Dropout add 1% relative performance in our model.

TABLE I
DATA OVERVIEW

Apps	Developer	Categories	Count
Baidu Map	Baidu.com	Travel & Local	6777
Baidu Post Bar	Baidu.com	Social	3234
Netease cloud music	Netease.com	Music & Audio	9888
iQIYI	iQIYI	Music & Audio	2634
JingDong	JD	Shopping	7956
Jinritoutiao	ByteDance	News & Magazines	5321
Meituan	Meituan.com	Lifestyle	12469
QQ	Tencent	Communication	1246
QQ music	Tencent	Music & Audio	1155
QQ reader	Tencent	Books & Reference	1563
Taobao	Taobao	Shopping	3431
Weibo	Sina	Social	3097
CTRIP	CTRIP	Lifestyle	2141
Zhihu	Zhihu.com	Social	2011
Tik Tok	Douyin.com	Social	7441
Ele.me	Ele.me	Lifestyle	16053
gtja	gtja.com	Finance	7734
QQ mail	Tencent	Tools	4879
Tencent News	Tencent	News & Magazines	5679
Alipay	Alipay.com	Finance	2301
Total	-	-	**107010**

IV. EXPERIMENT AND EVALUATION

A. Data Collection and Labeling

We create a traffic dataset based on the HTTPS protocol. The dataset contains more than 100,000 HTTPS flows of 20 popular applications. We execute an app both in several real Android devices and Android emulators. The app is automatically driven by the Android tool, monkeyrunner [21]. we execute the same app in different devices at one time and ensure there is no app running in the background by limiting the permissions of other applications in the android devices. We capture one app's traffic at a time to ensure it is ground truth. To avoid the influence of network environment, we kept capturing traffic by Wireshark [22] and tcpdump [23] in our laboratory in different time manually and automatically during a month.

B. Experimental Setup

AIBMF is based on TensorFlow [24] platform, and the training and testing process is carried out on a server. The server's hardware environment is as follows:

- CPU: 24 Intel (R) Xeon CPU E5-2650 v4 @ 2.2GHz
- GPU: 2 GTX TITAN X
- RAM: 128GB
- OS: Ubuntu 16.04 LTS

We divided the 100,000+ labeled examples into two parts: training set and test set. In order to ensure randomness, we performed shuffle operation on the data, in which the training set accounts for 90% and the test set accounts for 10%. The count of flows of each application is shown in Table I.

C. Evaluation Metrics

In order to make a reasonable and effective quantitative evaluation of the identification performance of AIBMF on the traffic data, this paper introduces some basic metrics: True

positives (TP), False negatives (FN), True negatives (TN), and False positives (FP). TP is the number of flows classified to app i exactly belong to app i. FN is the Number of flows classified to app i exactly belong to app j. TN is the Number of flows classified to app j exactly belong to app i. FP is the number of flows classified to app j exactly belong to app j.

In the training phase, the *Accuracy(ACC)* is used to indicate the improvement of the model identification ability, and the performance of the comprehensive display model in identifying HTTPS traffic is improved. The accuracy rate is the proportion of all the correct sample sizes to the training data during the iterative training, which is calculated according to formula (7).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

In the test phase, *Precision(P)* that shows how many flows predicted to app i are actual app i, *Recall(R)* that shows the percentage of flows that are correctly predicted versus all flows that belong to app i, and the comprehensive evaluation index *F1* value for HTTPS traffic are used as the evaluation criteria. *P* is calculated according to formula (8a); *R* is calculated according to formula (9a); the F1 value is a comprehensive evaluation index that combines two indicators, and is calculated according to formula (10a). To evaluate our method on all 20 apps, we introduce the average of P, R, F_1 — $Macro\ Precision$, $Macro\ Recall$, $Macro\ F_1$ as the overall metrics according to formula (8b), (9b), (10b).

$$P = \frac{TP}{TP + FP} \tag{8a}$$

$$Macro\ Precision = \frac{1}{20}\sum_{i=1}^{20} P_i \tag{8b}$$

$$R = \frac{TP}{TP + FN} \tag{9a}$$

$$Macro\ Recall = \frac{1}{20}\sum_{i=1}^{20} R_i \tag{9b}$$

$$F_1 = 2\frac{P \times R}{P + R} \tag{10a}$$

$$Macro\ F_1 = \frac{1}{20}\sum_{i=1}^{20} F_{1i} \tag{10b}$$

D. Key Parameters

1) Packet count of each flow: We collected 100,000+ flow data and studied the packet count distribution of various applications. As shown in Figure 4, the median of packets count of 100,000+ flows is 11, the packets count in a flow exceeds 16 only accounts for 32%, in our method, the packets count of each flow is set to 16 and the first 16 packets are taken in each flow.

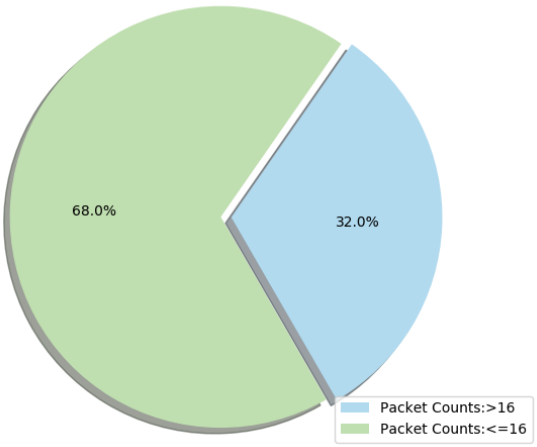

Fig. 4. Packet Counts of Each Flow

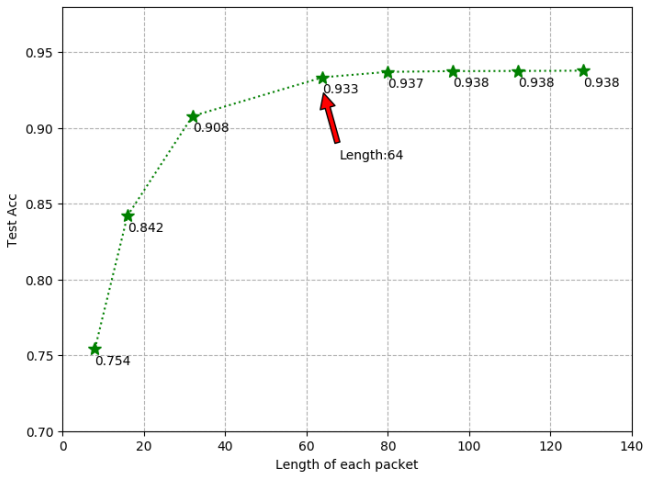

Fig. 5. Performance with Different Payload Length of Each Packet

2) Payload length of each packet: To decide how many bytes of each packet to use, we conducted a series of comparative experiments. As shown in Figure 5, each packet is intercepted with different lengths, as the intercept length increases, the recognition ability of the 1D-CNN is enhancing. When the first 64 bytes are taken, the accuracy is 93.34% on the test set. After the length of 64 bytes, the accuracy increases very slowly and the gains decrease with the length continues to increase, however the time complexity and space complexity of 1D-CNN will increase significantly, hence the length of each packet is set to 64.

E. Analysis and Evaluation of Experimental Results

1) Statistical features only: We extracted the statistical features using the open source tool— Scapy [15]. These features are based on the statistical characteristics of time and size. Such as maximum, minimum, average, variance, etc. as shown in Table II.

TABLE II
HTTPS STATISTICS

Statistics
Packet counts
TTL max
TTL min
TTL mean
TTL median
TTL var
packet_length max
packet_length min
packet_length mean
packet_length median
packet_length var
window max
window min
window mean
window median
window var
session_id_length max
client_extensions_length max
client_extensions_length min
client_extensions_length mean
client_extensions_length median
client_extensions_length var
client_ciphers counts
server_cipher

TABLE III
TRAFFIC IDENTIFICATION BASED ON STATISTICAL FEATURES ONLY

Machine learning algorithm	Precision	Recall	F1
Random Forest	0.8805	0.8172	0.8407
SVM-RBF	0.9214	0.5754	0.6676
DNN	0.7362	0.6293	0.6562
AIBMF	**0.918**	**0.909**	**0.913**

We use machine learning algorithms to perform experiments on 24-D statistical features, using Random Forest, SVM, and DNN algorithms. As shown in Table III, the experimental results indicate that using statistical features alone is not qualified for HTTPS traffic classification.

2) Analysis of model converging process in the training phase: We have two major comparisons in Figure 6. First, compared to 2D-CNN, 1D-CNN enhances the training accuracy by 4% with 40,000 training steps and converges faster. Second, compared to using 1D-CNN only, AIBMF model enhances the training accuracy by 2% with 40,000 training steps and converges faster.

3) AIBMF overall performance: Confusion matrix on the 10,000+ test samples is depicted in Figure 7. The identification performance is almost perfect where most of the mistakes are due to the share of network service among different apps especially for these developed by the same developer.

4) Model comparation: As shown in Figure 8, we compare AIBMF with the baseline [9] in our dataset. The evaluation results shows that AIBMF outperforms at least 2.0% on precision, recall and F1 score.

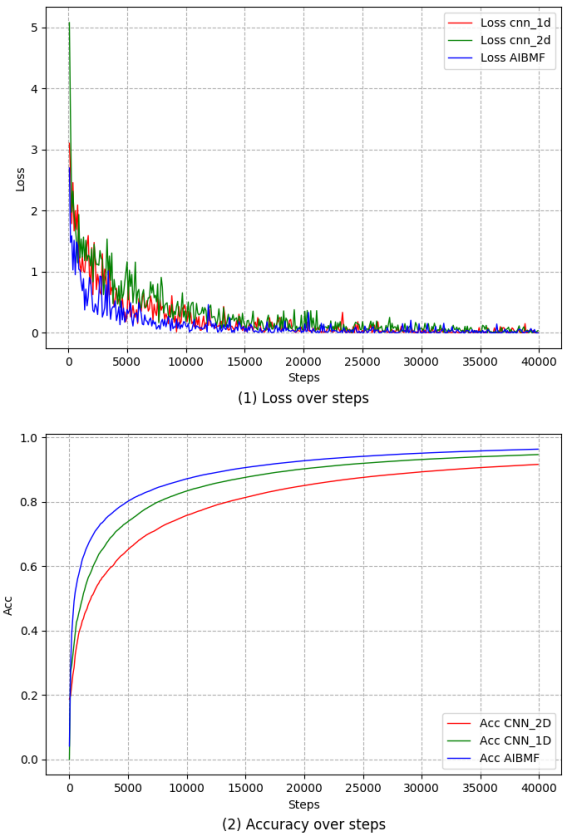

Fig. 6. Training Loss and Accuracy of three different deep learning models.

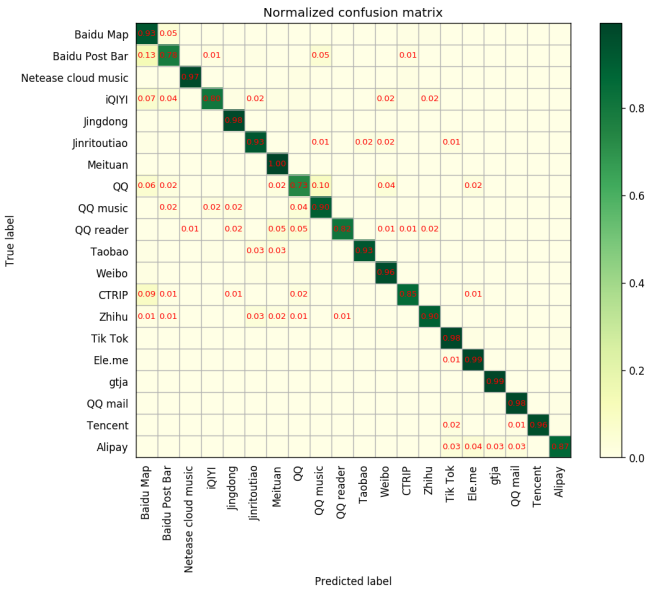

Fig. 7. AIBMF confusion matrix: For most applications, the recognition effect is very good. There are exceptions for the application developed by the same company. Some APPs provide their API to others services, such as Baidu map, which can also lead to confusion. The recognition effect of QQ software is the worst in our experiment. The may reason is that the communication protocol used by QQ is mainly UDP, supplemented by TCP protocol. We capture QQ traffic mainly from Qzone which mixes many other applications' traffic.

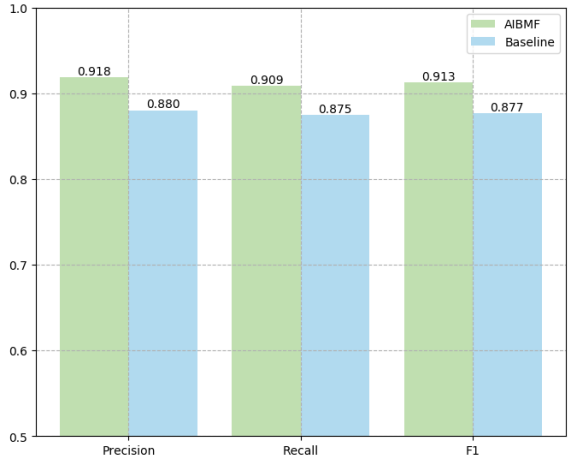

Fig. 8. AIBMF vs. Baseline. Note that the metrics of AIBMF are always higher than these of the baseline.

V. Discussion

The more complex the artificially defined statistical features are, the worse the versatility is in traffic identification tasks. However the identification ability is also limited only using deep learning to process payload ignoring the protocol-designed information. In fact some meaningful information defined by protocol such as content type costs little effort to obtain and is useful in traffic analysis. So analysing HTTPS traffic from multiple views and combining different features may be a powerful choice in the future encrypted traffic analysis.

VI. Conclusion And Future Work

This paper proposes a method—AIBMF for application identification over HTTPS traffic. Our approach has the ability to analyze HTTPS application traffic and associate HTTPS traffic flow with specific mobile apps. On the basis of this paper, the following work will be carried out:

(1) We plan to capture HTTPS traffic on other operating system platforms and validate feature extraction and traffic identification methods on the extended dataset.

(2) We tend to study other TLS Parameters and combine them with payload to improve our system's performance.

(3) We use the deep learning method for traffic identification in AIBMF, and the extracted features are very abstract. In the next stage, the explanatory of these features will be studied.

Acknowledgment

The research leading to these results has received funding from the National Natural Science Foundation of China (U1736128, No.61572496) and the National Key Research and Development Program of China (No.2016YFB0801502).

References

[1] Google, "Google transparency report," 2018.

[2] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[3] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, D. Ran, A. Dvir, and O. Pele, "Analyzing https encrypted traffic to identify user operating system, browser and application," 2016.

[4] J. Kohout and T. Pevnỳ, "Network traffic fingerprinting based on approximated kernel two-sample test," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 788–801, 2018.

[5] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order markov chains and application attribute bigrams," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

[6] J. Suárez-Varela and P. Barlet-Ros, "Towards accurate classification of https traffic in software-defined networks," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2018, pp. 1–6.

[7] W. M. Shbair, T. Cholez, J. Franois, and I. Chrisment, "Improving sni-based https security monitoring," in *IEEE International Conference on Distributed Computing Systems Workshops*, 2016, pp. 72–77.

[8] N. S. Yoav Nir, Rich Salz, "Transport layer security (tls) parameters," 2018.

[9] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *IEEE International Conference on Intelligence and Security Informatics*, 2017, pp. 43–48.

[10] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2005, pp. 50–60.

[11] C. V. Wright, F. Monrose, and G. M. Masson, "On inferring application protocol behaviors in encrypted network traffic." *Journal of Machine Learning Research*, vol. 6, no. 4, pp. 2745–2769, 2006.

[12] A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri, "Toward an efficient and scalable feature selection approach for internet traffic classification," *Computer Networks*, vol. 57, no. 9, pp. 2040–2057, 2013.

[13] Z. Chen, B. Yu, Y. Zhang, J. Zhang, and J. Xu, "Automatic mobile application traffic identification by convolutional neural networks," in *Trustcom/bigdatase/ispa*, 2017, pp. 301–307.

[14] NETRESEC, "SplitCap," accessed 2018-11-01. [Online]. Available: https://www.netresec.com/?page=SplitCap

[15] P. Biondi, "Scapy, 2012," accessed 2018-11-01. [Online]. Available: https://scapy.net/

[16] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[19] P. T. D. Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] A. Developers, "Monkeyrunner," 2015.

[22] P. Wu, "Wireshark.(2018)," *Retrieved April*, vol. 16, p. 2018, 2018.

[23] V. Jacobson, C. Leres, and S. McCanne, "Tcpdump public repository," *Web page at http://www. tcpdump. org*, 2003.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/