Smartphone Application Identification by Convolutional Neural Network

Shuang Zhao(✉) and Shuhui Chen

College of Computer, National University of Defense Technology, Changsha, China
Zhaos_abby@163.com, shchen@nudt.edu.cn

Abstract. Mobile traffic has received much attention within the field of network security and management due to the rapid development of mobile networks. Unlike fixed wired workstation traffic, mobile traffic is mostly carried over HTTP/HTTPS, which brings new challenges to traditional traffic identification methods. Although there have been some attempts to address this problem with side-channel traffic information and machine learning, the effectiveness of these methods majorly depends on predefined statistics features. In this paper, we presented an approach based on convolutional neural network without explicit feature extraction process. And owing to no payload inspection requirement, this method also works well even encrypted traffic appears. Six instant message applications are used to verify our approach. The evaluation shows the proposed approach can achieve more than 96% accuracy. Additionally, we also discussed how this approach performed under real-world conditions.

Keywords: Application identification
Convolutional neural network · Mobile traffic · Encrypted traffic
Network management

1 Introduction

With the proliferation of mobile devices and applications, the composition and diversity of network traffic changes tremendously. SmartInsights [1] points out that mobile devices dominate total minutes spent online and Apps drive dominant share of mobile time in all markets. China Statistical Report on Internet Development [2] reports that mobile internet users account for 96.3% of internet users in China. Consequently, mobile traffic has overtaken traditional workstation traffic and occupied a major portion of network traffic. Therefore, mobile traffic monitoring becomes a real concern for individuals, business and related organizations.

For instance, with the aid of mobile traffic identification technology, ISP could figure out which application is using the most bandwidth. Intrusion detection system could identify malicious traffic, and enterprises could identify and limit the use of related apps during office hours. It is worth mentioning that only passive traffic monitor is required during the analysis process.

L. Meng and Y. Zhang (Eds.): MLICOM 2018, LNICST 251, pp. 105–114, 2018.
https://doi.org/10.1007/978-3-030-00557-3_11

There are some problems when handling mobile traffic by traditional identification methods. Unlike traditional workstation traffic, mobile traffic has some special characteristics: (1) Mobile traffic is mostly carried over HTTP/HTTPS which indicates that port-based identification method nearly useless. (2) The effectiveness of DPI-based methods is diminishing because of encrypted HTTPS traffic. (3) Traditional workstation traffic has too coarse-grained identification targets such as protocols or services, while mobile traffic is required to match with Apps or even particular activities. (4) Technologies include clouding host, CDN (content distribution network), and third libraries also make server domain less reliable and the generated traffic more similar. (5) Mobile App markets grow fast, and Apps update frequently, so the identification method must be scalable. These features reveal that it is necessary to propose new identification methods for dealing with mobile traffic.

In recent years, some notable work has used machine learning and traffic patterns leaked through side-channel information such as packet sizes or time-related features to identify traffic [3–5]. These methods only employ traffic statistics features and do not involve payload, thus overcoming the problems raised by random ports and encryption techniques. However, the effectiveness of these methods depends on the handcrafted features heavily. On the one hand, it's difficult to extract abstract traffic features manually. On the other hand, the classifier's ability is limited by distinguishing all applications with the same set of features, given that different application has different distinguishable features. In addition, most features are complete flow related making real-time identification impossible.

To solve the mentioned problems, in this paper, we propose a real-time mobile traffic identification approach based on two-dimensional convolutional neural network (2D-CNN). This method only needs raw data as input without decrypting traffic. And 2D-CNN could extracts features automatically. The main process is as follows. The raw traffic is first separated into flows according to 5-tuple (SrcIp, DstIp, SrcPort, DstPort, protocol), and then only the application layer data of the first five packets with at least a byte of TCP data payload is reserved. Next, the data is transformed into a two-dimensional image as the 2D-CNN model' input. In the end, 2D-CNN model gives the prediction.

The main contributions of our work are as follows. Firstly, we present a real-time mobile traffic identification method based on 2D-CNN, which omits the feature extraction process and regardless of whether traffic is encrypted. Secondly, the experimental results show that the proposed method performs better than the state-of-the-art method Random Forest. Afterwards we also propose validation threshold to reduce the false positive rate. Finally, we discuss the influence of background traffic on model accuracy.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the proposed approach including data collection and model architecture. Section 4 presents results of the experiments and outlines the post-validation method. The influence of background traffic is discussed in Sect. 4.3. Section 5 concludes the paper.

2 Related Work

Current mainstream research of mobile traffic identification could be categorized into two categories. First one is to generate unique App signatures by tokens such as User-Agent field in HTTP requests. The Other one focuses on machine learning.

Xu et al. [6] used User-Agent field in HTTP requests to differentiate apps and analyze the resource usage. Dai et al. [7] automatically generated apps' fingerprints by domain name and HTTP request tokens. AppPrint [8] also discovered apps' signatures by parameters in URL or cookies. The limitation of these methods is that it only works for HTTP traffic which provides these specific tokens. Additionally, they also cannot distinguish between the traffic generated by the same third-party effectively.

Thanks to not needing to inspect the payload of traffic, machine learning-based identification method doesn't have the problems of the above work. Wang et al. [9] identified 13 apps by Random Forest with packets size and interval time features were used. Taylor et al. [10] applied Random Forest to identify 110 apps. They proposed burst modeling traffic and employed packet sizes as features. This work also attempted to identify ambiguous traffic that shared among apps. Hasan et al. [11] extracted the packet sizes patterns of the traffic produced by the first 20s when apps launched and identified thousands of apps. However, the features used in the above work have obvious concrete sense, whereas the more abstract features are overlooked.

As a kind of machine learning algorithm, the neural network has the advantage of extracting features automatically. A few impressive studies have applied CNN to classify mobile traffic. Chen et al. [12] encoded HTTP requests plain text and trained a 2D-CNN model to identify 20 apps. Deep Packet [13] employed stacked autoencoder and 1D-CNN to identify and characterize mobile traffic. Wang et al. [14] considered mobile traffic as a sequential data and therefore utilized 1D-CNN as the classification model. Work [15] comes closer to the work in this paper, which uses 2D-CNN to classify malware traffic. But we design a different processing method to generate the input data.

3 Methodology

3.1 Data Collection

To validate the proposed approach, a local dataset is collected by our lab members during one month. We selected 6 popular Android instant messaging applications, and captured these Apps' traffic by Tpacketcapture [16]. Then NetworkLog [17] is used to label the traffic accurately. Figure 1 depicts the procedure of data preprocessing.

Firstly, packets are split into flows according to the 5-tuple with the timeout set to 90s according to experience. Considering apps could use long connection, so a long connection would be separated into several flows due to the timeout limitation. We would retain flows which without SYN handshake, but flows

which have less than two packets with payload are removed. Secondly, flow is converted into an image which can be easily calculated in the 2D-CNN model. To identify a flow rapidly, we only employ first five payload carried packets' application layer data. The reason for retaining only the application layer data is that the link header does not have App-related information, and IP address and ports are also not reliable. Then, given that the MTU is mostly set to 1500 bytes, so each packet's payload can be represented as a $1 * 1500$ bytes vector. Zero padding and Truncation are used if needed. To preserve packet's direction information, the bytes from client to server are normalized to $[128, 255]$ and bytes in the other direction are normalized to $[0, 127]$. Finally, five packets are sequentially combined to a $5 * 1500$ 2D-vector, and converted to an $87 * 87$ image with normalized to range $[0, 1]$. The details of this dataset after preprocessing are shown in Table 1.

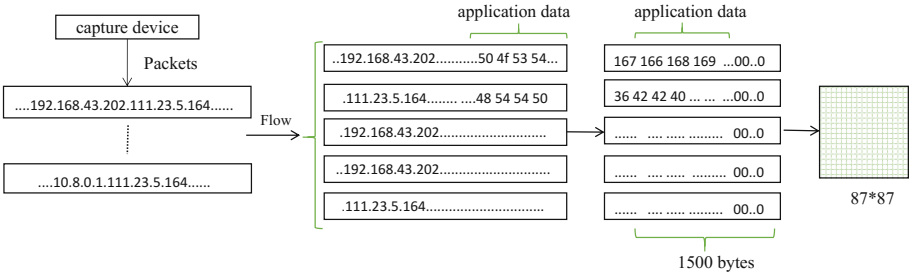


Fig. 1. The procedure of converting raw traffic to CNN input data

Table 1. The processed results of local dataset

Application	Flow size
ChaoXin	7185
DingTalk	5936
MoMo	8000
TanTan	7962
WeChat	7991
YiXin	7779

3.2 2D-CNN Architecture

2D-CNN has been successfully applied in the field of image processing. It mainly includes convolution and sampling layer, which can extract the abstract features within structured data and then identify the targets. After sufficient experiments and careful parameters tuning, the 2D-CNN model proposed in this paper is illustrated in Fig. 2, which contains two convolution layers and three full connection layers. Table 2 gives the detailed parameters of each layer.

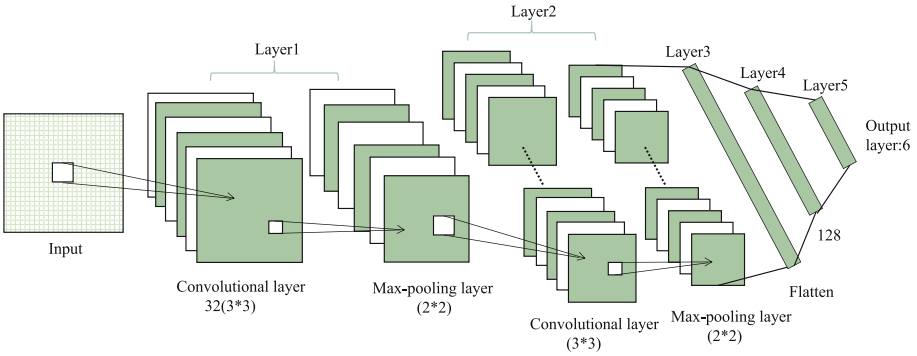


Fig. 2. The procedure of converting raw traffic to 2D-CNN input data

Table 2. Parameters of 2D-CNN model

a) Convolution layer parameters

Layer	Activation Function	Filter	Max-pooling	Dropout
1	ReLU	32(3*3)	2*2	/
2	ReLU	64(3*3)	2*2	0.25

b) Full connection layer parameters

Layer	Node	Activation Function	Dropout
3	Flatten	ReLU	/
4	128	ReLU	0.25
5	6	Softmax	/

4 Experiments and Evaluations

We use the mentioned local dataset to evaluate our approach and compare with the results of the stat-of-the-art method Random Forest. 80% of the dataset are randomly sampled as training set and the rest is test set.

2D-CNN model is implemented using Keras library [18] with TensorFlow as backend. Training batch size is set to 64, loss function and optimizer use cross entropy and Adadelta optimizer built in Keras. The final model is obtained after 100 epochs.

Random Forest is implemented by data mining tool Weka [19]. Like the input data preprocessing of 2D-CNN, at most first five payload carried packets are used to extract 37 statistical features for each flow, as listed in Table 3. Since

Random Forest itself has the function of feature selection, no feature selection is performed. The final model contains 100 trees and the rest of the parameters remain the default of Weka.

Table 3. Features used in random forest

Feature type	Description	Number
Port	Port number in unidirection	2
Packet size	First three payload carried packets size in unidirection. Min, Max, Mean, Std Dev of payload carried packets size in unidirection and flow	18
Inter-packet Time	Max, Min, Mean, Std Dev of inter packet time in unidirection and flow	12
Packets	Packets number transferred in unidirection. The ratio of the payload carried packets transferred in unidirection	4
Bytes	The ratio of volume bytes transferred in two directions	1

4.1 Local Dataset Evaluation

Four evaluation metrics including recall, precision, F-measure and overall accuracy are used. The results show that the overall accuracy of 2D-CNN model is 96.94%, and Random Forest is 96.08%. Tables 4 and 5 display the experimental results of 2D-CNN model and Random Forest. Table 6 shows the confusion matrix of two models.

From the above Tables, we can observe that the performance of 2D-CNN in each class higher than that of Random Forest, which indicates 2D-CNN could extract more abstract and effective features.

Table 4. Results of 2D-CNN model

Apps	Precision	Recall	F-measure
ChaoXin	97.22%	95.21%	96.20%
DingTalk	91.79%	97.10%	94.37%
MoMo	98.67%	97.56%	98.11%
TanTan	97.69%	98.55%	98.11%
WeChat	98.27%	96.61%	97.43%
YiXin	96.94%	96.44%	96.69%
Mean	96.76%	96.91%	96.83%

Table 5. Results of random forest model

Apps	Precision	Recall	F-measure
ChaoXin	95.9%	94.4%	95.1%
DingTalk	92.5%	95.1%	93.8%
MoMo	97.2%	96.8%	97.0%
TanTan	96.8%	97.7%	97.2%
WeChat	97.5%	96.4%	96.9%
YiXin	95.7%	95.7%	95.7%
Mean	95.93%	96.02%	95.95%

Table 6. Confusion matrix of 2D-CNN model/random forest model

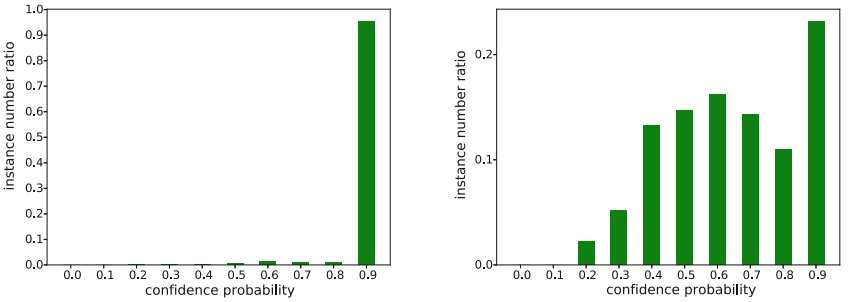
True	Pred					
	ChaoXin	DingTalk	MoMo	TanTan	WeChat	YiXin
ChaoXin	1331/1320	42/37	3/9	11/16	3/3	8/13
DingTalk	10/25	1140/1117	3/7	3/5	7/5	11/15
MoMo	5/6	10/14	1561/1548	11/14	3/9	10/9
TanTan	3/8	6/5	6/7	1566/1552	6/8	2/9
WeChat	5/3	19/14	7/11	7/10	1537/1533	16/20
YiXin	15/15	25/20	2/10	5/7	8/15	1490/1478

4.2 Post-validation

It's worth noting that the last layer in the CNN model uses softmax as the activation function, i.e., softmax function is used to output the probability of each class. Therefore, the output of the last layer could represent the confidence of the prediction.

From this perspective, we calculate the probability distribution of the predictions on the local data set as shown in Fig. 3. We can see that above 90% of true positive instances have a confidence value higher than 0.9. On the contrary, most false predictions have lower confidence value. Thus, we could use confidence threshold to further confirm the prediction. If confidence is lower than the threshold, CNN model could refuse to give a prediction.

We set the confidence threshold to 0.85, and retest the test set on the 2D-CNN model. The experiment shows that the average precision is 99.15%, average recall is 99.11%, and average F-measure is 99.13%. Due to the rejection of the sample with low confidence, flow coverage decreases to 93.67%. However, this trade-off is desirable for the scenarios where concentrate on accuracy or app coverage rather than flow coverage.



(a) Distribution of true positive (b) Distribution of false prediction

Fig. 3. Confidence probability distribution

4.3 Influence of Background Traffic

In real world, network traffic is dominated by the background traffic rather than the target traffic. We examine the impact of background traffic on classifier performance in this section. Based on the original output classes, we add a class Other, and all non-target traffic belongs to Other. Therefore, the classifier can handle background traffic.

Then we add 52853 background flows to the mentioned local dataset and retrained the 2D-CNN model. The confidence threshold is configured to 0.85. The results show that the average precision, recall and F-measure are 97.69%, 97.71%, 97.7% respectively. And the flow coverage is 93.64%. The confusion matrix is shown in Fig. 4.

From the above experimental results, it can be concluded that under the influence of background traffic, the performance of the classifier will decrease. Furthermore, it can be speculated that the accuracy of the classifier will become lower as time goes on because the background traffic set used in the training phrase is incomplete. Therefore, it is necessary to continuously monitor the classifier's performance and update the model with new traffic samples.

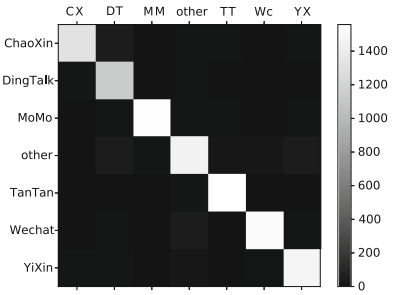


Fig. 4. Confusion matrix of 2D-CNN with identifying other class

5 Conclusion

Traditional traffic identification methods have been unable to handle mobile traffic effectively. In this paper, we have studied the efficiency of 2D-CNN model to address the mobile traffic identification. Our results proved that 2D-CNN is an effective approach and superior to the state-of-the-art method Random Forest. In addition, we found that the 2D-CNN model can satisfy the requirement of the scenario where focus on accuracy simply by setting a confidence threshold. At the end of the paper, we also discussed the impact of background traffic on classifier performance. As future work we will continue to further study how to handle background traffic and automatically identify new traffic classes.

References

1. Mobile Marketing Statistics compilation. https://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/
2. The 40th China Statistical Report on Internet Development. http://cnnic.cn/hlwfzyj/hlwxzbg/hlwtjbg/201708/P020170807351923262153.pdf
3. Gerard, D., Arash, L., Mamun, M., Ali, G.: Characterization of encrypted and VPN traffic using time-related features. In: The International Conference on Information Systems Security and Privacy, Italy, pp. 94–98 (2016)
4. Zhang, J., Chen, X., Xiang, Y., Zhou, W.-L., Wu, J.: Robust network traffic classification. J. IEEE/ACM Trans. Netw. **23**(4), 1257–1270 (2015)
5. Taylor, V., Spolaor, R., Conti, M., Martinovic, I.: AppScanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: IEEE Symposium on Security and Privacy, pp. 439–454 (2016)
6. Xu, Q., Ermanet, J., Gerber, A., Mao, Z., Pang, J., Venkaraeaman, S.: Identifying diverse usage behaviors of smartphone apps. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, Berlin, pp. 329–344 (2011)
7. Dai, S.-F., Tongaonkar, A., Wang, X.-Y., Nucci, A., Song, D.: NetworkProfiler: towards automatic fingerprinting of Android apps. In: Proceeding IEEE INFOCOM, Italy, pp. 809–817 (2013)
8. Miskovic, S., Lee, G.M., Liao, Y., Baldi, M.: AppPrint: automatic fingerprinting of mobile applications in network traffic. In: Mirkovic, J., Liu, Y. (eds.) PAM 2015. LNCS, vol. 8995, pp. 57–69. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15509-8_5
9. Wang Q.-L., Yahyavi, A., Kemme, B., He, W.-B.: I know what you did on your smartphone: inferring app usage over encrypted data traffic. In: Communications and Networking Symposium, pp. 433–441 (2015)
10. Taylor, V., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. J IEEE Trans. Inf. Forensics Secur. **13**, 63–78 (2018)
11. Alan, F., Kaur, J.: Can android applications be identified using only TCP/IP headers of their launch time traffic? In: Wireless Network Security, pp. 61–66 (2016)
12. Chen, Z.-Y., Yu, B.-W., Zhang, Y., Zhang, J.-Z., Xu, J.-D.: Automatic mobile application traffic identification by convolutional neural networks. In: Trustcom/bigdatase/ispa, pp. 301–307(2017)

13. Lotfollahi, M., Zade, R., Siavoshani, M., Saberian, M.: Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. arXiv (2017)
14. Wang, W., Zhu, M., Wang, J.-L., Zeng, X.-W., Yang, Z.-Z.: End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: IEEE International Conference on Intelligence & Security Informatics, pp. 43–48. IEEE Press, Beijing (2017)
15. Wang, W., Zhu, M., Zeng, X.-W., Ye, X.-Z., Sheng, Y.-Q.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking, pp. 712–717. IEEE Press, Da Nang (2017)
16. TPacketCapture. https://play.google.com/store/apps/details?id=jp.co.taosoftware.android. Packetcapture
17. Pragmatic Software, Network Log. https://play.google.com/store/apps/details?id=com.googlecode.networklog
18. Keras: The Python Deep Learning library. https://keras.io/
19. Weka 3: Data Mining Software in Java. https://www.cs.waikato.ac.nz/ml/weka/