MAAN: A Multiple Attribute Association Network for Mobile Encrypted Traffic Classification

Fengzhao Shi[1,2,3], Chao Zheng[1,3(✉)], Yiming Cui[1,2,3], and Qingyun Liu[1,2,3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
zhengchao@iie.ac.cn
[2] School of Cyberspace Security, University of Chinese Academy of Sciences, Beijing, China
[3] National Engineering Laboratory for Information Security Technology, Beijing, China

Abstract. With the rapid development of mobile applications and the rising concern over user privacy, cryptographic protocols, especially Secure Socket Layer/Transport Layer Security (SSL/TLS), are widely used on the Internet. Many networking and security services call for application-level encrypted traffic classification before conducting related policies. Exiting methods exhibit unsatisfying accuracy using the partial handshake information or only the flow-level features. In this paper, we propose a novel encrypted traffic classification method named Multiple Attribute Associate Network (MAAN). MAAN is a unified model that automatically extracts features from handshake messages and flows. Moreover, the MAAN has acceptable time consumption and is suitable to apply in real-time scenarios. Our experiments demonstrate that the MAAN achieves 98.2% accuracy on a real-word dataset (including 59k+ SSL sessions and covering 16 applications) and outperforms the state-of-the-art methods.

Keywords: Encrypted traffic classification · SSL/TLS · Handshake messages · Application Data · Network management

1 Introduction

Mobile encrypted traffic classification, which classifies mobile encrypted traffic into specific applications, plays a very significant role in many areas, such as Quality of Service (QoS), network management, intrusion detection, and prevention systems [26]. Taking intrusion detection as an example, network administrator needs to identify the applications before analyzing user behavior. Hence, it has gained significant interest in both academia and industry.

Traditional port-based and payload-based traffic classification methods are inappropriate for mobile encrypted traffic classification because of encryption and dynamic port technology [16]. Machine learning methods, which don't need

N. Park et al. (Eds.): SecureComm 2020, LNICST 335, pp. 172–189, 2020.
https://doi.org/10.1007/978-3-030-63086-7_11

port and payload information, are suitable to solve this task. Although many excellent methods have been proposed, there still remain several problems. Firstly, traditional machine learning methods need sophisticated feature engineering, and model optimization cannot optimize the feature extraction process [19]. Secondly, traditional machine learning is difficult to convert all TLS handshake messages into feature vectors, thus missing part of handshake message features [10]. Thirdly, some networking and security services need early prediction results to conduct related policies before the transmission of communication payloads. However, the previous methods are unable to solve this problem effectively based on the characteristics of a single flow. For example, current methods using sequence features require complete flow information, which leads these methods to get results at the end of the flow [16,18,19,26,27].

In this paper, we try to apply some deep learning methods to overcome the above challenges. Deep learning methods have drawn widespread attention in many areas, such as speech recognition, visual object recognition, machine translation, etc. Some deep neural networks show impressive performance in dealing with natural language. Since the handshake messages can be seen as a kind of character language, we can use deep learning methods for mobile encrypted traffic classification [17]. Motivated by the above idea, we propose a novel approach named the Multiple Attribute Associate Network (MAAN). MAAN takes a hierarchical neural network to extract message-level and flow-level features from handshake messages and flows. Moreover, MAAN achieves an acceptable time delay for real-time classification.

Our contributions can be briefly summarized as follow:

- We propose a unified MAAN model for mobile encrypted traffic classification. MAAN automatically learns representative features from raw encrypted traffic and doesn't need manual feature engineering.
- MAAN combines handshake messages with flows to extract message-level and flow-level features. In a real-world dataset, MAAN achieves satisfactory results and outperforms several state-of-the-art methods. Meanwhile, MAAN performs an acceptable time delay and is suitable for real-time classification.

2 Background

2.1 SSL/TLS Basics

The Secure Socket Layer (SSL) protocol and its successor Transport Layer Security (TLS) protocol are the encryption standards of two largest mobile markets, the App Store and the Google Play [1,2]. As a result, most of the applications use the SSL/TLS to communicate with servers, which leads to a high proportion of the SSL/TLS traffic in mobile networks. Figure 1 shows an example of the SSL/TLS protocol communication session. It consists of two processes that are the handshake process and the data transfer process. In the handshake process, the client first sends Client Hello to the server and try to initiate a connection.

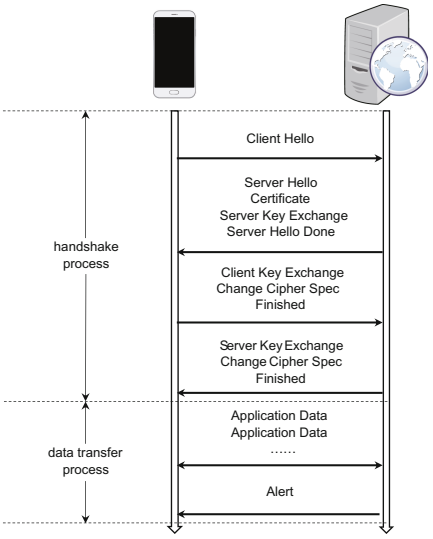

Fig. 1. Example of the SSL/TLS Protocol Communication Session

Then, the server sends Server Hello and Certificate to the client for authentication and encryption parameter negotiation. Next, the two sides use four kinds of messages, Server Key Exchange, Client Key Exchange, Change Cipher Spec, and Finished Message to agree on a master secret and complete handshake. In the data transfer process, the two sides transfer Application Data, which has been encrypted with a master secret. Finally, the session terminates with an Alert Message from the server. Moreover, SSL/TLS uses session reuse to reduce resource consumption during the handshake phase. For brevity, we refer to SSL/TLS as SSL in this paper.

2.2 Related Work

We have briefly stated the drawbacks of convolutional traffic classification methods, the port-based and the payload-based, in the introduction. The merits and defects of the machine learning methods for encrypted traffic classification will be elaborately discussed below. For a clear comparison, we categorize the related work into two classes: statistical machine learning methods and deep learning methods.

Statistical Machine Learning Methods: Statistical machine learning methods contain two processes that are feature engineering and classifier selection. At present, researchers mainly focus on feature engineering. We summarize three kinds of features widely used in the encrypted traffic classification: statistical features, flow sequential features, and original message features.

Statistical Features: Statistical features are used to solve encrypted traffic classification problems combined with traditional machine learning methods, such as random forest, SVM. Taylor et al. [30] proposed AppScanner, which uses the packet lengths or some statistical features of packet lengths as features and takes the SVM or random forest as the classifier. To get the best results, AppScanner uses the classification validation technique. Then, Taylor et al. optimized AppScanner by the technique of ambiguity detection [31]. Al-Naami et al. [7] extracted statistic features from the packet, uni-burst and bi-burst (e.g., packet length, uni-burst size). They also propose a new method to overcome concept drift. Hayes et al. [14] generate a more robust web fingerprint from some statistical features by random forest. And Shi et al. [28] take a deep learning method to optimize the statistical features and enhance the performance of traffic classification. However, these methods require manual feature designing that is mainly based on rich experiences, professional knowledge, and lots of human effort.

Flow Sequential Features: Message type sequence and packet length sequence are two widely used flow sequential features. Korczyński et al. [16] first use a message type sequence to establish a one-order Markov chain for application traffic classification. Shen et al. [26] incorporate attribute bigrams into the second-order Markov chain to solve the misclassification problem of the one-order Markov chain. Liu et al. [18] combine message type sequence one-order Markov chain with Length Block sequence one-order Markov chain to generate probability feature vectors and use the random forest as the classifier. Due to the need for adequate sequence information, these methods are hard to classify encrypted traffic in real-time.

Original Message Features: These methods take advantage of the differences in SSL protocol. Sengupta et al. [25] exploit the diversity in SSL implementations for application traffic classification. Whereas, this method is unsuitable for real-time classification due to the high computational complexity of feature extraction. Chen et al. [10] take Domain Name, Common Name, Organization, and Application Data lengths as the features and combine the contextual reference to improve the performance of classification. However, some features of the method are difficult to extract in practice.

Deep Learning Methods: The deep learning methods combine feature engineering and classifier optimization into a unified model. These methods mainly contain two categories: packet-based methods and flow-based methods.

Packet-Based Methods: These methods take raw packet as input and design deep neural network for classification. Wang et al. [35] achieve protocol identification andanomalous protocoldetection with DNN and SAE. Datanet [34] designs three

neural networks for encrypted traffic classification. Zeng et al. [37] combine three neural networks to adapt to different network environments. Lotfollahi et al. [21] adopt the stacked autoencoder and one-dimensional convolutional neural network to extract features from packet payload. Li et al. [17] transform the raw network datagram into several segments and use a hierarchical attention encoder to extract features. However, these methods only focus on the packet level features and do not consider the flow level information.

Flow-Based Methods: Yang et al. [36] adopt autoencoder and CNN with manual extraction features for encrypted traffic classification. However, the method doesn't take advantage of the automatic feature extraction of the neural network, and it can't be applied in real-time classification by reason of features' limitation. Aceto et al. [5] propose MIMETIC, a new architecture that can incorporate multiple inputs and multiple models to improve the performance of classification. Whereas, the MIMETC requires a through design which cannot ignore network expertise's help. And Liu et al. [19] adopt autoencoder to establish an end-to-end model that learns features from the raw flow sequences. This method is no more suitable for real-time classification than the flow sequential features methods mentioned above.

3 Architecture of MAAN

MAAN extracts features from encrypted traffic flow, which contains a complete SSL session. According to the SSL protocol, handshake messages and Application Data messages contain application identity information. For handshake messages, we consider Client Hello, Server Hello, and end-entity certificate, which is the final certificate signed for servers, as inputs. For Application Data messages, the message lengths of Application Data reflect the application layer's transmission logic, which can be used to classify. Therefore, we use the message lengths of first N Application Data, named as Application Data lengths, as inputs.

MAAN is a hierarchical model that consists of 5 layers, as shown in Fig. 2. In our method, a segment preprocessor first breaks three handshake messages into three integer sequences. Next, the message feature extractor transforms three integer sequences and Application Data lengths to $N+3$ message-level representation vectors. Then, a flow feature extractor extracts a flow-level representation vector from $N+3$ message-level representation vectors. After this, a dense layer compresses the flow-level representation vector and creates the final fingerprint. Finally, the fingerprint of the flow is introduced to a classification layer to predict the class of the flow. Herein, the details of each layer are as follows.

3.1 Segment Preprocessor

Segment preprocessor is designed to convert three handshake messages into three integer sequences (named as S_{client}, S_{server} and $S_{certificate}$) that can

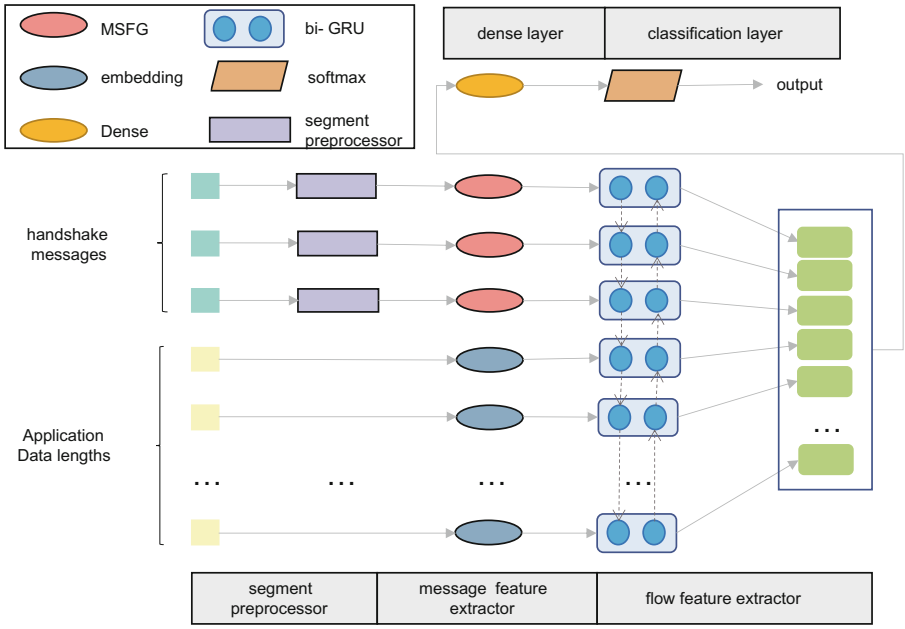

Fig. 2. The architecture of MAAN

be calculated by the deep learning models. We refer to each handshake message as a datagram. Firstly, segment preprocessor cuts the datagram into 2-byte chunks. Then, every chunk is transformed into an integer between 1 to 65536. Besides, each type of datagram is transferred into a sequence with fixed length (S_{client} and S_{server} are set to 300, and $S_{certificate}$ is set to 5000). In the case of sequences longer (resp. shorter) than the considered fixed-length data formats, these sequences are truncated (resp. padded with zeros) to the designed length. To be specific, for these session-resumed flows, we set the end-entity certificate to a sequence of zero.

3.2 Message Feature Extractor

The message feature extractor takes three integer sequences (S_{client}, S_{server}, $S_{certificate}$) and Application Data lengths (named as S_{length}) as the inputs, extracting message-level features.

We embed each element of the S_{length} to a vector via embedding layer [23]. The embedding layer is a lookup table essentially. For the element $s_i, i \in [1, N]$ in S_{length}, the corresponding embedding vector of s_i is the s_i-th row of matrix $E \in \mathbb{R}^{k \times d}$, where E, k, and d represent the embedding matrix, the size of S_{length}'s element set and the dimension of embedding vectors respectively. The embedding matrix E is trainable and contains the context information of s_i [23]. Finally, we can obtain N representation vectors $[l_1, l_2, ..., l_N]$ where $l_i = E_{s_i}$.

In this way, $l_i, i \in [1, N]$ can get more information for classification and boost the classification performance [19].

Due to the local correlation of handshake messages and the speed of message feature extraction, we extract handshake message features with CNN. The specific structure is as follows: three integer sequences are fed into the same architecture named message sequence feature generator (MSFG) based on CNN. The architecture of MSFG is shown in Fig. 3. The sequence is first fed into an embedding layer similar to S_{length}. Because of the embedding layer's ability to retrieve sequence context information, MSFG can extract more field information in this layer. Then, we extract deeper features with a 1D-CNN like [15]. The 1D-CNN consists of a one-dimensional convolution layer, a global max-pooling layer, and a dense layer. To extract more features, we use two kernel sizes to make the dual-channel. The dense layer is a fully connected layer and calculated as follow:

$$m_d = Selu(Wm + b) \tag{1}$$

where m represents the output of the max-pooling layer, $Selu$ is Activation function, W and b are weight and bias of fully connected layer respectively. The MSFG converts S_{client}, S_{server} and $S_{certificate}$ to three representation vectors m_{client}, m_{server} and $m_{certificate}$.

Finally, we can obtain the message representation vector sequence $M = [m_1, m_2, ..., m_{N+3}]$ where m_1, m_2, m_3 denote $m_{client}, m_{server}, m_{certificate}$ respectively and $m_j, j \in [4, N+3]$ represent $l_i, i \in [1, N]$.

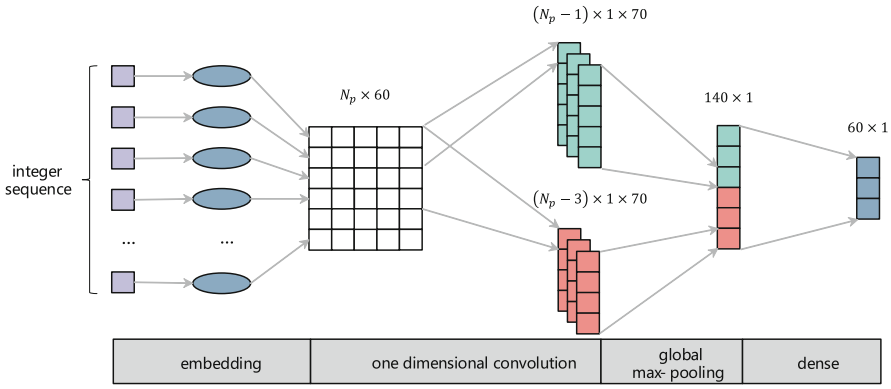

Fig. 3. The Architecture of MSFG.N_p represent integer sequence length (Subscript p represents Client Hello, Server Hello and end-entity certificate).

3.3 Flow Feature Extractor

The network flow conforms to the sequence characteristics with time correlation, so RNN is suitable for extracting flow-level features. Due to the vanishing gradient problem [9] and lacking backward information of traditional RNN, we use

the bidirectional GRUs (bi-GRU) in this layer. Meanwhile, we use single bi-GRU to achieve acceptable time consumption. Given a message representation vector sequence $M = [m_1, m_2, ..., m_{N+3}]$, the bi-GRU contains a forward GRU network $\overrightarrow{GRU}$ which reads M from m_1 to m_{N+3} and a backward GRU network $\overleftarrow{GRU}$ which reads M from m_{N+3} to m_1:

$$\overrightarrow{h}_t = \overrightarrow{GRU}(\overrightarrow{h}_{t-1}, m_t), t \in [1, N+3] \tag{2}$$

$$\overleftarrow{h}_t = \overleftarrow{GRU}(\overleftarrow{h}_{t+1}, m_t), t \in [1, N+3] \tag{3}$$

where $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are the forward and backward hidden states respectively, and the initial hidden state vectors $\overrightarrow{h}_0$ and $\overleftarrow{h}_{N+4}$ are both zero vectors. Then, the bi-GRU conbines the forward hidden state $\overrightarrow{h}_t$ and the backward hidden state $\overleftarrow{h}_t$ to create a two direction outputs like: $o_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$. Finally, we concatenate the all output o_t to obtain the flow based feature vector z as follows:

$$z = [\overrightarrow{h}_1, \overleftarrow{h}_1, \overrightarrow{h}_2, \overleftarrow{h}_2, ..., \overrightarrow{h}_t, \overleftarrow{h}_t], t \in [1, N+3] \tag{4}$$

3.4 Dense Layer

To compress the dimension of flow-based feature z and overcome the risk of over-fitting, we take a dense layer similar to Eq. 1 ($tanh$ is Activation function in this layer) and create a higher representation of the entire flow, named as z_c.

3.5 Classification Layer

The flow fingerprint z_c is input into a softmax classifier to generate the probability distribution over different applications:

$$p(i) = \frac{exp(\theta_i z_c + b_i)}{\sum_{a=1}^{A} exp(\theta_a z_c + b_a)} \tag{5}$$

where $p(i)$ is the probability that z_c is application $i (i \in [1, A])$. With the distribution, we take the application with the maximum probability as the prediction label.

4 Experiment

In this section, we introduce our experiments to evaluate the MAAN. All experiments are conducted on a PC with an Intel Xeon E5-2682V4 CPU, Nvidia Tesla P4 GPU, and 16 GB RAM running Ubuntu 18.04. We conduct all experiments with python.

4.1 Dataset

To the best of our knowledge, active dataset collection and passive dataset collection are widely used data collection schemas. Considering that passive dataset collection is hard to label comprehensively, we only use active dataset collection schema in this paper. Meanwhile, we take a manual collection schema to reflect the users' real usages. In this scheme, an iOS phone connects to a laptop's Wifi network and only run one application at a time. When the iOS phone is running an application, the laptop captures all the flows generated by this application using Wireshark [4], and the operator labels them with the application name. Then, we extract the SSL encrypted traffic by matching 443 port. Some flows generated by iOS system don't belong to any app in dataset. To get a purer dataset, we filter out these flows by pattern matching, which uses two pattern strings (*.icloud.* and *.apple.*) to match SNI values.

With the collection scheme above, we collect over 59 thousands encrypted traffic flows corresponding to 16 applications from January 1, 2020 to January 20, 2020. Table 1 shows detailed information about each application.

Table 1. Overview of dataset

Application	Flows	Packets	Bytes
Alipay	4298	219035	374 MB
Amap	4084	81486	252 MB
Ele	3825	484892	559 MB
Baidu Tieba	3978	459833	528 MB
Baidu Map	4025	281585	299 MB
Zhihu	4349	814797	1019 MB
Meituan	4066	273988	311 MB
Weibo	3636	774925	957 MB
Youdao	3872	684298	888 MB
Linkin	2091	528350	550 MB
Douyin	3899	1235634	1586 MB
Jd	4375	1294364	1593 MB
Booking	2567	115650	105 MB
Airbnb	2027	327338	374 MB
Evernote	4165	644703	798 MB
Bilibili	3808	211843	211 MB
Total	59065	8432721	10094 MB

4.2 Experiment Setting

Setting of MAAN: We take Client Hello, Server Hello, end-entity certificate, and the message lengths of the first 10 Application Data as the inputs (the quantities of Application Data messages will be discussed in Sect. 4.4). The dimension of the embedding layer is set as 60. We set the kernel sizes of the convolutional network to 2 and 4, and the filter of each kernel to 70. We set the dimension of hidden states of each GRU as 50. Moreover, we take a dropout with 0.1 ratios to avoid over-fitting, and we set epochs to 10. Our model is implemented with TensorFlow [3]. We detail the parameters selection in Appendix A.

Cross Validation: To obtain a reliable and stable model, we establish 5-fold cross-validation. More specifically, we split the total dataset into five parts, and every time four parts are used for training while one part is used for testing. All the process repeats five times with different parts. Finally, we use the average value of five-fold cross-validation results as the final result.

Assessment Criteria of Model: We evaluate all the methods with three metrics, Precision (Prec.), Recall (Rec.), and Accuracy (Acc.). For app a, Precision is the proportion of real encrypted flows belonging app a in the encrypted flows classified as app a, while Recall is the proportion of encrypted flows correctly classified as app a in the total app a encrypted flows [10]. Accuracy is the proportion of encrypted flows correctly classified in total flows.

4.3 Comparisons with Existing Approaches

Comparative Experiments: We use three state-of-the-art methods as comparison experiments. The description and setup of these experiments are shown below:

- **MaMMF:** MaMMF [18] constitutes probability feature vectors with the message type Markov chain and Length Block Markov chain, taking Random Forest as the classifier. We set Length Block to cover 90% of the whole packets.
- **MAAF-SFF:** MAAF [10] takes Domain Name, Common Name, Organization, and Application Data lengths as features. Meanwhile, it uses the context reference to improve the classification performance. Some features of MAAF are hard to be used in practice (e.g., Domain Name is confusing to associate with the corresponding flow). Therefore, we only consider the single flow's characteristics (Common Name, Organization, and Application Data lengths) in this method and abandon the context reference. We call it as MAAF-Single Flow Features (MAAF-SFF). We set the parameters same as the paper and take XGBoost as the classifier.
- **FS-Net:** FS-Net [19] uses an encoder-decoder structure to extract features from raw flow sequences and classify them with neural network. We use the packet length sequence as the FS-Net' input and set the parameters same as the paper.

Table 2. Experimental results of different approaches

Application	MaMMF		MAAF-SFF		FS-Net		MAAN	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Alipay	0.8719	0.8328	0.9394	0.8954	0.9471	0.9523	**0.9830**	**0.9540**
Amap	0.9297	0.9699	0.9910	0.9705	0.9822	**0.9851**	**0.9962**	0.9850
Ele	0.8400	0.7038	0.9091	0.9161	0.9347	0.9361	**0.9466**	**0.9836**
Baidu Tieba	0.8210	0.8326	0.8791	0.7387	0.9068	0.9305	**0.9829**	**0.9765**
Baidu Map	0.8912	0.8301	0.8213	0.8473	0.9408	0.9188	**0.9911**	**0.9749**
Zhihu	0.7217	0.9218	0.9788	0.9399	0.9803	**0.9826**	**0.9976**	0.9821
Meituan	0.8783	0.9059	0.9842	0.9712	0.9753	0.9893	**0.9857**	**0.9904**
Weibo	0.8621	0.7015	0.8312	0.9561	0.9532	0.9587	**0.9862**	**0.9944**
Youdao	0.8084	0.8574	0.8951	0.8894	0.9723	0.9412	**0.9853**	**0.9906**
Linkin	0.7379	0.6161	0.8806	0.8130	0.9322	0.8603	**0.9710**	**0.9327**
Douyin	0.8570	0.8525	0.8836	0.8494	0.9536	0.9578	**0.9743**	**0.9850**
Jd	0.6937	0.9177	0.8724	0.8734	0.9752	0.9708	**0.9828**	**0.9924**
Booking	0.9529	0.7613	0.7966	0.8766	0.9790	0.9378	**0.9842**	**0.9862**
Airbnb	0.6746	0.8661	0.6984	0.8432	0.8369	0.9789	**0.9662**	**0.9856**
Evernote	0.9522	0.9430	0.9357	0.9774	**0.9916**	0.9839	0.9850	**0.9988**
Bilibili	0.8655	0.6096	0.9667	0.9414	0.9786	0.9700	**0.9847**	**0.9847**
Accuracy	0.8308		0.8984		0.9557		**0.9822**	

Results: Table 2 shows the experimental results of different approaches. We can obtain the following conclusions:4

1) MAAN outperforms all the other methods. According to Table 2, MAAN obtains the best precision performances on 15 of 16 applications except for Evernote, but its recall is the best. For the overall performance, MAAN achieves the best performance on accuracy. In comparison with other methods, MAAN's advantages are credited to the multiple features association and the unified learning structure.

2) Handshake messages have many representative features that can improve the accuracy of classifier. MaMMF, FS-Net, and MAAN all extract flow-level features, and our method outperforms the other methods, according to Table 2. MAAN benefits from the handshake messages, which contain much plaintext of application identity.

3) The unified framework leads MAAN to achieve a better performance than the two-stage method. Our MAAN outperforms the MAAF-SFF, although they all use handshake messages and Application Data lengths as inputs. The MAAF-SFF needs to extract features from the handshake messages manually, so it is challenging to make use of all the handshake information. By contrast, MAAN

uses a unified structure that can automatically extract comprehensive handshake information. Meanwhile, MAAF-SFF ignores the flow-level based features.

4.4 Analysis of MAAN

Segment Lengths: To convert the handshake messages into the integer sequences that can be calculated by neural networks, we must segment handshake messages with a fixed length. There are two viable segment lengths, 1-byte and 2-byte. In this experiment, we focus on the effectiveness of two segment lengths for classification. Table 3 shows the accuracy of two segment lengths. We figure out that the accuracy of 2-byte segment was 0.94% higher than that of 1-byte segment. We believe the reason is that 1-byte segment divides all the fields of messages, making it difficult for the classifier to extract features. Meanwhile, 2-byte segment can produce a shorter integer sequence than 1-byte segment, which can improve classification speed.

Table 3. Comparison of 1-byte and 2-byte segment

Segment length	1-byte	2-byte
Accuracy	0.9728	0.9822

Combination of Messages: This experiment studies the contribution of each handshake message and the collaboration between the handshake messages. We traverse all combinations of the three handshake messages and study the classification accuracy for each combination. Based on the experimental results, we can optimize the combination of the handshake messages to suit the actual network scenario.

The experimental results are presented in Fig. 4. Due to the limited space, we replace the message names by abbreviations. We use CL, SH and EC to represent Client Hello, Server Hello and end-entity certificate respectively. We use ALL to represent the combination of three messages.

According to the experimental results in Fig. 4, three handshake messages improve the classification accuracy, which verifies the original intuition that handshake messages can effectively classify applications. In detail, Client Hello contributes the most information for classification, while Server Hello contributes the least. We think the reason for this situation is that Server Hello contains fewer protocol parameters than Client Hello. Furthermore, we found that Client Hello and end-entity certificate can be harmoniously combined to improve the accuracy significantly.

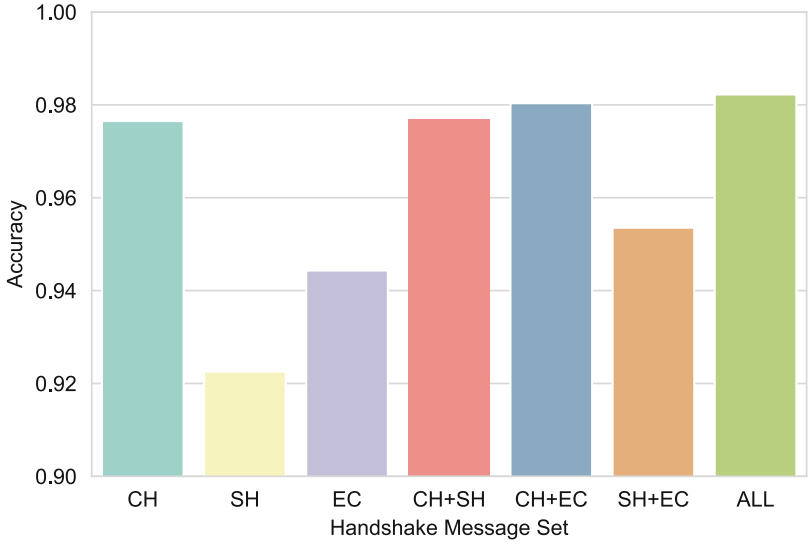

Fig. 4. Clssification Accuracy under Different Handshake Message Sets

Quantities of Application Data Messages: This experiment studies the classification accuracy under different quantities of Application Data messages. We iterate over the quantities of Application Data messages, from zero to twenty, with an interval of two. Figure 5 shows the upward trend of the classification accuracy with the increase of the quantities of Application Data messages. Obviously, as the quantities of Application Data messages increase, the accuracy is rising, too. When the quantities of Application Data messages increase to 10, the accuracy reaches its maximum and does not increase afterwards. Therefore, we consider 10 to be the optimal value for the quantities of Application Data messages.

4.5 The Efficiency of MAAN

In this section, we study the ability of MAAN for real-time classification. We first illustrate online classification ability of MAAN by comparing with other approaches. Then, we discuss the performance of MAAN in a real-time scenario.

Table 4 lists the comparison results of different approaches. The classification time of single flow in MAAN is 1.5 ms, which is 10 times faster than that of MaMMF and FS-Net. Although the classification time of MAAN is longer than that of MAAF-SFF, MAAN has a five-fold reduction of error rate compared to MAAF-SFF. Therefore, we believe that MAAN is more suitable for online classification than other methods.

In real-time scenarios, the classifier is expected to classify the traffic as soon as possible. MAAN can obtain earlier predictions with fewer Application Data messages. Because handshake messages contain many identity information, MAAN

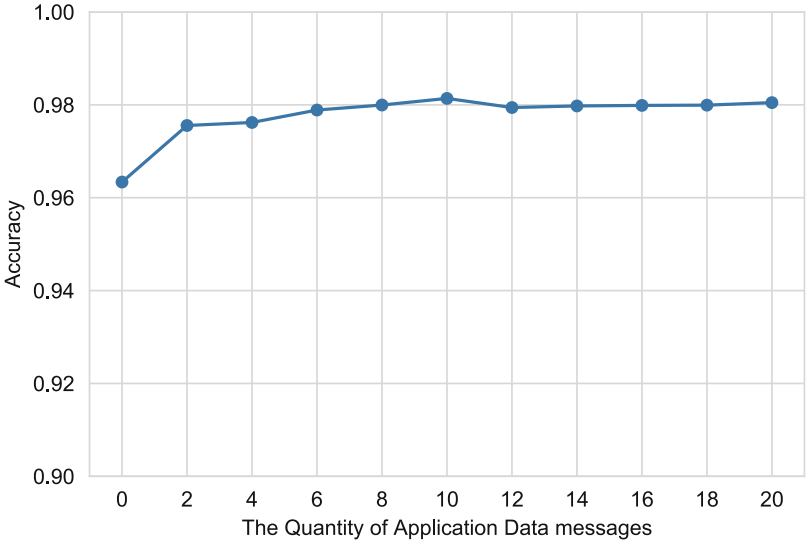

Fig. 5. Classification Accuracy under Different Quantities of Application Data messages

can achieve 96.34% accuracy with zero Application Data messages, which outperforms other methods.

Table 4. Efficiency of Approaches. Classification Time of Single Flow is the ratio of test time to total number of test samples. Error Rate is the ratio of misclassified simples to all.

Approaches	MAAN	MaMMF	FS-Net	MAAF-SFF
Classification Time of Single Flow	1.5 ms	19.9 ms	19.0 ms	0.1 ms
Error Rate	1.8%	16.9%	4.4%	10.2%

5 Discussion

In this section, we will discuss how to apply our method in other encryption protocol. Firstly, most encryption protocols exchange keys with a handshake protocol, which is plaintext and contains many identity information (e.g. encryption parameters), so we can take the plaintext handshake messages as the first part of the input. Then, the message length field cannot be obtained in some encryption protocols, we can substitute it with burst, which is the sum of the data packets's lengths in the same direction. Finally, we can build a model with similar architecture to MAAN for classification.

Although we can apply MAAN to other protocols through the above process, there still exist several problems: firstly, many encryption protocols do not contain authentication information in the plaintext of the handshake (e.g. ssh), which will greatly affect the performance of the classifier. Secondly, MAAN will obtain poor performance for the protocols that contains little plaintext (e.g. TLS1.3). We will find more suitable encrypted traffic classification methods to solve the above problems in our future work.

6 Conclusion

In this paper, we propose a Multiple Attribute Association Network (MAAN), which is an efficient encrypted traffic classification method based on deep learning. MAAN combines handshake messages with flows to improve classification accuracy. Besides, it establishes a unified model that doesn't require feature engineering. We evaluate the effectiveness of MAAN with network traffic collected from the real-world. According to the experimental results, MAAN achieves 98.2% accuracy on the real-world dataset and outperforms three state-of-the-art approaches. Moreover, the experiments demonstrate the effectiveness of the attributes employed by MAAN and reveal that MAAN is suitable for real-time classification.

A Parameters Selection

We use grid search method to find the optimal parameters. The specific results are shown in the Table 5.

Table 5. Parameters selection

Architecture	Parameter	Range	Step	Optimal parameter
Message feature extractor	embedding dimension	$[10, 100]$	10	60
	kernal size	$2, 3, 4, 5$		$2, 4$
	CNN activation	$Relu, Selu, tanh, None$		$None$
	CNN filters	$[10, 100]$	10	70
	Dense activation	$Relu, Selu, tanh$		$Selu$
Flow feature extractor	GRU hidden states	$[10, 100]$	10	50
	GRU activiation	$Relu, Selu, tanh$		$tanh$
Dense layer	dense dimension	$[50, 200]$	50	100
	dense activiation	$Relu, Selu, tanh$		$tanh$
	optimizer	$Adam, RMSProp$		$Adam$
	dropout	$[0.05, 0.3]$	0.05	0.1

References

1. App transport security. https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33
2. Network security configuration. https://developer.android.com/training/articles/security-config
3. Tensorflow. https://www.tensorflow.org/
4. Wireshark. https://www.wireshark.org/
5. Aceto, G., Ciuonzo, D., Montieri, A., Pescapè, A.: MIMETIC: mobile encrypted traffic classification using multimodal deep learning. Comput. Netw. **165**, 106944 (2019)
6. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges. IEEE Trans. Netw. Serv. Manage. **16**(2), 445–458 (2019)
7. Al-Naami, K., et al.: Adaptive encrypted traffic fingerprinting with bi-directional dependence. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 177–188 (2016)
8. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1723–1732 (2017)
9. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Networks **5**(2), 157–166 (1994)
10. Chen, Y., Zang, T., Zhang, Y., Zhouz, Y., Wang, Y.: Rethinking encrypted traffic classification: a multi-attribute associated fingerprint approach. In: 2019 IEEE 27th International Conference on Network Protocols (ICNP), pp. 1–11. IEEE (2019)
11. Conti, M., Mancini, L.V., Spolaor, R., Verde, N.V.: Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Inf. Forensics Secur. **11**(1), 114–125 (2015)
12. Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., Ghorbani, A.A.: Characterization of encrypted and VPN traffic using time-related. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), pp. 407–414 (2016)
13. Fu, Y., Xiong, H., Lu, X., Yang, J., Chen, C.: Service usage classification with encrypted internet traffic in mobile messaging apps. IEEE Trans. Mob. Comput. **15**(11), 2851–2864 (2016)
14. Hayes, J., Danezis, G.: k-fingerprinting: a robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 1187–1203 (2016)
15. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
16. Korczyński, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 781–789. IEEE (2014)
17. Li, R., Xiao, X., Ni, S., Zheng, H., Xia, S.: Byte segment neural network for network traffic classification. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2018)

18. Liu, C., Cao, Z., Xiong, G., Gou, G., Yiu, S.M., He, L.: MaMPF: encrypted traffic classification based on multi-attribute Markov probability fingerprints. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2018)
19. Liu, C., He, L., Xiong, G., Cao, Z., Li, Z.: FS-Net: a flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1171–1179. IEEE (2019)
20. Liu, J., Fu, Y., Ming, J., Ren, Y., Sun, L., Xiong, H.: Effective and real-time in-app activity analysis in encrypted internet traffic streams. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 335–344 (2017)
21. Lotfollahi, M., Siavoshani, M.J., Zade, R.S.H., Saberian, M.: Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft. Comput. **24**(3), 1999–2012 (2020)
22. Mandic, D.P., Chambers, J.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. Wiley, New York (2001)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
24. Razaghpanah, A., Niaki, A.A., Vallina-Rodriguez, N., Sundaresan, S., Amann, J., Gill, P.: Studying TLS usage in android apps. In: Proceedings of the 13th International Conference on emerging Networking Experiments and Technologies, pp. 350–362 (2017)
25. Sengupta, S., Ganguly, N., De, P., Chakraborty, S.: Exploiting diversity in android TLS implementations for mobile app traffic classification. In: The World Wide Web Conference, pp. 1657–1668 (2019)
26. Shen, M., et al.: Classification of encrypted traffic with second-order Markov chains and application attribute bigrams. IEEE Trans. Inf. Forensics Secur. **12**(8), 1830–1843 (2017)
27. Shen, M., Wei, M., Zhu, L., Wang, M., Li, F.: Certificate-aware encrypted traffic classification using second-order Markov chain. In: 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2016)
28. Shi, H., Li, H., Zhang, D., Cheng, C., Cao, X.: An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. Comput. Netw. **132**, 81–98 (2018)
29. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1928–1943 (2018)
30. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: AppScanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 439–454. IEEE (2016)
31. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. IEEE Trans. Inf. Forensics Secur. **13**(1), 63–78 (2017)
32. Velan, P., Čermák, M., Čeleda, P., Drašar, M.: A survey of methods for encrypted traffic classification and analysis. Int. J. Network Manage. **25**(5), 355–374 (2015)
33. Wang, P., Chen, X., Ye, F., Sun, Z.: A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access **7**, 54024–54033 (2019)

34. Wang, P., Ye, F., Chen, X., Qian, Y.: DataNet: deep learning based encrypted network traffic classification in SDN home gateway. IEEE Access **6**, 55380–55391 (2018)
35. Wang, Z.: The applications of deep learning on traffic identification. BlackHat USA **24**(11), 1–10 (2015)
36. Yang, Y., Kang, C., Gou, G., Li, Z., Xiong, G.: TLS/SSL encrypted traffic classification with autoencoder and convolutional neural network. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 362–369. IEEE (2018)
37. Zeng, Y., Gu, H., Wei, W., Guo, Y.: $deep - full - range$: a deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access **7**, 45182–45190 (2019)
38. Zhang, J., Chen, X., Xiang, Y., Zhou, W., Wu, J.: Robust network traffic classification. IEEE/ACM Trans. Networking **23**(4), 1257–1270 (2014)
39. Zou, Z., Ge, J., Zheng, H., Wu, Y., Han, C., Yao, Z.: Encrypted traffic classification with a convolutional long short-term memory neural network. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 329–334. IEEE (2018)