Journal of Network and Computer Applications 166 (2020) 102711

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Encrypted traffic classification based on Gaussian mixture models and Hidden Markov Models

Zhongjiang Yao [a,b], Jingguo Ge [a,b,**], Yulei Wu [c,*], Xiaosheng Lin [d], Runkang He [a,b], Yuxiang Ma [e]

[a] *Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[b] *School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*
[c] *College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK*
[d] *Jilin Province Academy of Environmental Sciences, Jilin, China*
[e] *School of Computer and Information Engineering, Henan University, Kaifeng 475004, China*

A R T I C L E I N F O

Keywords:
Traffic classification
Encrypted traffic
Gaussian mixture model
Hidden Markov model

A B S T R A C T

To protect user privacy (e.g., IP address and sensitive data in a packet), many traffic protection methods, like traffic obfuscation and encryption technologies, are introduced. However, these methods have been used by attackers to transmit malicious traffic, posing a serious threat to network security. To enhance network traffic supervision, this paper proposes a new traffic classification model based on Gaussian mixture models and hidden Markov models, named MGHMM. To evaluate the effectiveness of the proposed model, we first classify protocols and identify the obfuscated traffic by experiments. Then, we compare the classification performance of MGHMM with that of the latest Vector Quantiser-based traffic classification algorithm. On the basis of the experiment, the relation between the classification and the number of hidden Markov states, and the number of mixture of Gaussian distributions required to describe the hidden states, are analyzed.

1. Introduction

With the fast development of network monitoring systems, such as firewalls and intrusion detection systems, traffic generated by users can be easily monitored by a third party, e.g., Internet service providers and service providers. The third party can even infer the user's behavior and intention by analyzing network traffic. For example, the authors (Queiroz et al., 2019) proposed that the network monitors the utilization of network resources by user traffic. To protect personal data, users adopt many traffic protective measures, such as using encrypted transmission protocols (e.g., SSL/TLS) to protect their sensitive data and using anonymous communication networks (e.g., Tor (Dingledine et al., 2004)) to hide their IP address.

However, traffic protective measures are currently being used by malicious users to protect and disguise their attacks. The growth in the use of SSL/TLS encompasses both legal and malicious activities, e.g., criminals rely on legitimate SSL certificates to spread their malicious content. On the Zscaler cloud platform (Desai, 2018), an average of 8.4 million malicious activities based on SSL/TLS are blocked every day. In addition, Matthew Prince, CEO of a cloud security and services company CloudFlare, said on the company's blog that 94% of Tor network traffic is malicious (Prince, 2016). The fact that Tor currently integrates traffic obfuscation techniques (e.g., Meek, obfs4) fuels malicious activities. Therefore, traffic classification, especially for the regulation of traffic that imposes protective measures, is of paramount importance.

Traffic classification is one of the most important traffic engineering methods. It uses the inherent characteristics of network traffic, such as packet size and packet arrival interval, to identify the types of traffic. Traditional traffic classification methods are mainly to distinguish the types of protocols (such as P2P) or applications (such as BitTorrent) that the traffic is dependent on. In recent years, network security has received more attention for the identification of encrypted traffic, which lead to rapidly progresses in research of traffic classification. Traffic protection measures such as network traffic obfuscation techniques, have been constantly introduced into the Internet, bringing big challenges to efficient traffic classification.

* Corresponding author.
** Corresponding author. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
E-mail addresses: yaozhongjiang@iie.ac.cn (Z. Yao), gejingguo@iie.ac.cn (J. Ge), Y.L.Wu@exeter.ac.uk (Y. Wu), linxiaosheng@163.com (X. Lin), herunkang@iie.ac.cn (R. He), yma@henu.edu.cn (Y. Ma).

https://doi.org/10.1016/j.jnca.2020.102711
Received 19 December 2019; Received in revised form 23 March 2020; Accepted 13 May 2020
Available online 21 May 2020

Traditional traffic classification mainly identifies traffic through protocol ports, such as the port 21 for FTP and the port 80 for HTTP. However, many applications currently use dynamic port negotiation mechanisms, making port-based methods no longer applicable (Karagiannis et al., 2005). Deep packet inspection (DPI)-based approaches have been considered to be effective and reliable for unencrypted traffic. However, the number of encrypted Internet traffic is significantly increasing, and many applications are using protocol encapsulation or obfuscation techniques to circumvent network monitoring policies (Karagiannis et al., 2004). Therefore, DPI-based methods are no longer effective.

A statistical-based approach analyzes traffic attributes firstly, such as information entropy and the number of connections. A machine learning (ML) based algorithm is then used to classify traffic based on the statistical attributes of these features. For example, the authors in Dong and Jain (2019) proposed a machine learning method based on a Bayesian model to identify encrypted Skype traffic, which also can be used to obfuscate other traffic. However, many traffic protective measures usually use encryption or padding techniques to interfere traffic statistics. Thus, the performance of existing methods based on feature statistics is low, and there is high rate of miss detection and false detection.

Based on the feature selection and extraction method in the paper (Liu et al., 2019a), this paper investigates existing traffic protective measures and finds that existing traffic generation relies on a single traffic protection policy, which cannot be changed once implemented as it is introduced in the form of a plug-in and is cold-started. The unchanged protection policy makes the Inter-Packet Time (IPT) and Packet Size (PS) interference rule fixed. That is to say, the statistical characteristics of IPT and PS are basically stable even if some techniques are called to randomize or regularize the packet attribute values. Meanwhile, the analysis of Tan et al. (2019) considers that features-varying behavior patterns, such as IPT, can imply traffic content information, so the combination of these features will inevitably make the traffic more detailed. Therefore, to improve the ability of monitoring network traffic, this paper proposes a new traffic classification mechanism by using the state features formed by a two-dimensional observation vector that is consisted of IPT and PS. It is useful for efficient traffic classification if the traffic protective measure has a single encryption policy or obfuscation strategy.

The main contributions of this paper can be summarized as follows:

- To establish the connection between IPT and PS, we propose to introduce the parameters of the covariance matrix of a mixture of Gaussians (MOG) to associate IPT and PS, which can effectively improve the accuracy of the flow feature description.
- This paper proposes a novel traffic classification model - Mixture of Gaussians based Hidden Markov Model (MGHMM). For the first time, it merges MOG with Hidden Markov Model (HMM) for traffic classification. The model uses MOG to depict two-dimensional observation probability density of each state. It then constructs the HMM model by considering the state changes of traffic during the communication.
- MGHMM is evaluated using the traffic dataset collected from a lab-built local area network and a publicly available traffic dataset. Extensive experimental results verify the effectiveness of the proposed MGHMM model for traffic classification from three levels, i.e., traffic protocol, traffic content, and application.
- We compare MGHMM with the latest Vector Quantiser-based traffic classification algorithm in terms of their classification performance on mainstream traffic protocols. The experimental results show the superiority of our proposed model.
- We propose a new method for selecting model parameters. By fitting F_1 performance metric and the computing cost with the variation curve of the number of hidden states and the number of mixture of Gaussians, we present a method for efficiently selecting the optimal model parameters.

The rest of the paper is organized as follows. Section 2 describes the motivation of our work and related work. Section 3 presents the MGHMM model, including the model inputs and the detailed derivations. In Section 4, the validity of MGHMM in traffic classification is verified by extensive experiments. It is compared with the latest classification algorithm. The corresponding parameter optimization method is also proposed. Section 5 gives the conclusion of this paper.

2. Related work

The current traffic protective measures can resist the port-based rules of firewalls and intrusion detection systems through dynamic port methods, and encryption algorithms can be used to get rid of the filtering of DPI systems. Researchers have therefore highlighted the need to move away from port and DPI based approaches and focus on statistical methods of traffic classification (Tahaei et al., 2020).

Encrypted traffic classification is still a research hotspot in recent years, and new research results are constantly being produced. According to the summary of Wang et al. (2019), the classification of encrypted traffic usually uses deep learning methods, such as convolutional neural network (CNN) (Rezaei and Liu, 1812) and long short-term memory (LSTM) (Zou et al., 2018). In addition, new methods have been proposed. For examples, Casino et al. (2019) proposed a method called HEDGE to classify encrypted and compressed packets using a threshold-based method. FS-net was an end-to-end classification model in Liu et al. (2019b) for encrypted traffic classification, and MIMETIC used multimodal deep learning for classifying mobile encrypted traffic as proposed in Aceto et al. (2019). The above methods either have complex feature extraction, or are difficult on model training. They are therefore not conducive to generalization.

The protected traffic is often not only encrypted but also obfuscated. The obfuscation technology makes the encrypted traffic no difference from unencrypted traffic by tunneling or regularization, but it is still encrypted in essence. This is achieved by making the traffic payload being randomized and the other features of the traffic are disturbed. However, the IPT and PS are the only features to make statistical analysis work for a given protective measure. IPT and PS have been widely used for traffic classification and identification. An early application flow identification classifier which relied on the first 4 ∼10 packets'- sizes of an observation sequence, was proposed in Wright et al. (2004). The authors proposed two separate classifiers using IPT and PS in Wright et al. (2004) and Auld et al. (2007), respectively. An extension of Wright et al. (2004) was proposed in Georgoulas and Moessner (2010), in which they presented a complex state structure for each packet and proposed a method trying to join IPT and PS through vector quantisation with a codebook. A packet-level traffic classifier managed to join IPT with PS in Korczynski and Duda (2014), but these two features were still computed separately within the same function.

HMM has been proposed for traffic classification and identification. Wright et al. (2004) proposed two HMM models using IPT and PS separately. A traffic classifier that is based on multiple Packet-Level Hidden Markov Models (PL-HMMs) and uses Gamma function to characterize IPT and PS probabilities density, was proposed in Dainotti et al. (2008). Its idea of combining multiple modular traffic classifiers is similar to Aceto et al. (2018). Although PL-HMMs can be used to identify different application traffic and obtain high traffic classification accuracy, it is computationally expensive as multiple PL-HMMs are required. Although Chen et al. (2016) constructed a single HMM model based on Tor network virtual circuit construction process and Tor log information, it can only effectively identify Tor flows which contain Tor circuit initial messages and have not been obfuscated by obfuscation technologies like Meek. Similarly, Korczynski and Duda (2014) used the random fingerprint of SSL/TLS-based application traffic in different applications to construct HMM to identify different application traffic, but

Table 1
The comparison of existing recent traffic classification algorithms.

Algorithm	Input	Advantages	Disadvantages
CNN (Rezaei and Liu, 1812)	packet-level data	good at classifying application traffic	difficult in model training
HEDGE (Casino et al., 2019)	three features from randomly sampled subset traffic	high accuracy	difficult in feature extraction
FS-net (Liu et al., 2019b)	raw data	end-to-end traffic classification	difficult in feature extraction and model training
MIMETIC (Aceto et al., 2019)	traffic object segmentation	can overcome performance limitations	difficult in feature extraction
LSTM (Zou et al., 2018)	packet-level data	analysis of feature timing	difficult in model training
PL-HMM (Wright et al., 2004)	packet-level data	reliable classification performance	high computation overhead
WENC (Pan et al., 2017)	packet-level data	reliable classification performance	poor adaptability to new scenarios
Hidden semi-Markov model (Huang et al., 2017)	packet-level data	simple to implement	lack of verification of real data
hybrid DNN-HMM (Tan et al., 2019)	flow-level data	fine-grained ability to identify load types	complicated to implement

it needs a large amount of observation and analysis work in the early stage. The most recent studies on traffic classification based on HMM were conducted in Pan et al. (2017), Huang et al., 2017 and Tan et al. (2019). Weighted ENsemble Classifier (WENC) based on HMM, which studied the HTTPS handshake process and the following data transmission period, was proposed for classification of HTTPS encrypted traffic in Pan et al. (2017). The authors in Huang et al., 2017 proposed a Long-Term Evolution (LTE) signal detection scheme based on Hidden semi-Markov model to capture the spatial-temporal characteristics of normal wakeup packet generation behavior, but the experiments were based on simulation rather than real network environment. A hybrid DNN-HMM model proposed in Tan et al. (2019) identify content types based on the GMM-HMM captured underlying time-varying behavior patterns, in which it does not need to inspect the external protocols or applications.

After analyzing the above encrypted traffic classification methods (a summary is presented in Table 1), we propose MGHMM, an HMM based on MOG to describe the traffic fingerprint by using two-dimensional probability density distribution of the observation formed by IPT and PS. MOG introduces a mixing factor that can more accurately characterize the density distribution of IPTs and PSs, including the links between IPTs and PSs. The hidden states of observation and their transition probability form a fully connected state topology. An observation sequence maps a hidden state transition sequence, whose occurrence probability can be used as a discriminant index. MGHMM is a single HMM and needs fewer features and computational overhead for traffic classification.

This journal paper is an extended version of a previously published conference paper (Yao et al., 2018). This journal version further expands on the following aspects:

- The investigation of classic and latest studies of traffic classification are added, which enriches related work.
- More detailed derivation process of MGHMM is presented. In the derivation process, we show the fitting effects on states and Gaussian mixture distributions for IPT logarithm and PS separately to prove that our model is well constructed.
- We add experimental results on traffic classification for protocol-level traffic with MGHMM using UNB open dataset. The results show that our model not only can be used to identify obfuscated traffic but also can classify general protocol traffic.
- We compare the performance of the proposed MGHMM model with the existing VQ-based traffic classification algorithm in identifying protocol traffic and obfuscating traffic. The experimental results show the superiority of the proposed scheme.
- We propose a new method for selecting model parameters. By fitting F_1 performance metric and the computing cost with the variation curve of the number of hidden states and the number of mixture of Gaussians, we present a method for efficiently selecting the optimal model parameters.

3. The proposed MGHMM model

In this section, we first introduce the proposed model and its detailed derivations, and then present the training method for the model.

3.1. The proposed MGHMM

In order to get rid of the limitation of a specific type flow and classify target traffic f from background traffic, this paper presents the MGHMM model. MGHMM takes the two-dimensional observation vectors composed of IPT and PS as inputs. In MGHMM, the observations are clustered as hidden states. Each hidden state cluster is characterized by MOG, and the hidden state transition rule is characterized by the Markov chain. The occurrence probability of the observation sequence then outputs.

Specifically, MGHMM is composed of a series of hidden states and observation sequence. The hidden state variable s_i is contained in the state set $S = \{s_1, \dots, s_i, \dots, s_N\}$, where N is the sum of the states. The observable variable r_l is contained in the observation sequence $R = \{r_0, \dots, r_l, \dots, r_L\}$. Each observation in R is $r_l = \{\log(ipt_l), ps_l\}^T$, where $\log(ipt_l)$ is the logarithm of IPT between the lth packet and $(l + 1)$th packet and ps_l is the PS of the lth packet. The two variables of ipt_l and ps_l can be correlated with a mixing coefficient. As MGHMM is based on MOG and HMM, both the MOG and HMM are characterized with their own parameters. Therefore, MGHMM can be characterized by a set of parameters as:

$$\lambda = \{\varepsilon, A, w, m, \Phi\} \tag{1}$$

where ε is a prior probability for hidden states. A is the state transition matrix, whose element a_{ij} $(1 \le i \le N, 1 \le j \le N)$ is the probability of transition from state s_i to state s_j. w is the vector of mixture coefficient, also called weight, whose element w_{ib} is the bth Gaussian distribution in the state s_i. m is the mean vector, whose element is the mean μ_{ib} of the bth Gaussian. Φ is the full covariance matrix of Gaussian distribution, whose element is σ_{ib}.

Each observation sub-sequence corresponds to a hidden state sequence of HMM. Each state is characterized by a bank of B Gaussian as follows:

$$\hbar_i(R) = \sum_{b=1}^{B} w_{ib} N(R; m, \Phi) \tag{2}$$

where $N(R; m, \Phi)$ is the bth Gaussian distribution of state s_i. In what follows, the MOG and HMM in the construction of the proposed MGHMM are elaborated.

3.1.1. Mixture of Gaussians (MOG)

To characterize a hidden state, we introduce MOG. Each state has an MOG combined with several different Gaussians with different weights. Each Gaussian computes a two-dimensional variable with a covariance matrix, which joins IPT and PS. The two-dimensional probability density of the HMM state estimated by MOG is the basis of the HMM's assessment of state transition probabilities.

The Expectation-Maximization algorithm (Dempster et al., 1977) is a well-founded statistical algorithm to get around this problem by an iterative process. In order to obtain the optimal parameter value of MOG, this paper introduces the Expectation-Maximization algorithm, where the expectation step (E-step) computes the expected probability of an observation coming from a specific Gaussian component with the current estimate for the parameters, and the maximization step (M-step) computes parameters maximizing the expected log-likelihood found in the E-step. In the training phase, the observations in every flow are clustered into N states, and the observation sub-sequence $\{r'_0, \ldots, r'_l\}$ is re-ordered by time stamp after being extracted as state s_i from the original observation sequence.

E-step: we first choose a state s_i from the clustered observations, and each state consists of B Gaussian components. Then, we estimate the probability of the lth observation in the training observation sequence with prior probability, which is generated by the bth Gaussian component.

$$\eta_{lib} = \frac{\omega_{ib} N(R; m, \Phi)}{\sum_{b=1}^{B} w_{ib} N(R; m, \Phi)} \tag{3}$$

M-step: After E-step, it needs to re-determine the parameters of each Gaussian component to maximize the Gaussian parameters to obtain the optimal values. The mixed Gaussians parameters mainly contain mean vector m and covariance matrix Φ, and its weight in state s_i.

The weight of a Gaussian is:

$$\widehat{\omega}_l = \frac{\sum_{l=1}^{L} \eta_{lb}}{B} \tag{4}$$

The mean vector can be expressed as:

$$\widehat{\mu}_l = \frac{\sum_{l=1}^{L} \eta_{lb} r_l}{\sum_{l=1}^{L} \eta_{lb}} \tag{5}$$

The covariance matrix of a Gaussian distribution is:

$$\widehat{\sigma}_l^2 = \frac{\sum_{l=1}^{L} \eta_{lb} (r_l - \mu_{lb})(r_l - \mu_{lb})^T}{\sum_{l=1}^{L} \eta_{lb}} \tag{6}$$

Repeat the E-step and M-step to improve the parameters of MOG until the parameters go convergence.

3.1.2. Hidden Markov Model (HMM)

Different traffic have different state clusters and different transitions among states, which can be used as the traffic fingerprint. After the estimation of the parameters of MOG, we compute the HMM parameters of MGHMM with the Expectation-Maximization algorithm, where E-step computes the transition probability between any two HMM states and M-step re-computes the MGHMM parameters.

E-step: After determining each hidden state of HMM, E-step begins to calculate the probability of occurrence of each state. The probability of an HMM state is estimated with the Forward-Backward algorithm (Fan, 2001). The Forward-Backward algorithm is adopted because it is a prevalent and efficient algorithm to compute transition probability among HMM states.

The Forward variable is:

$$\alpha_j^{l+1} = \sum_{j=1}^{N} \alpha_j^l \xi_{ji} f_i(r_{l+1}) \tag{7}$$

The Backward variable can be given by:

$$\beta_i^l = \sum_{j=1}^{N} \xi_{ij} f_j(r_{l+1}) \beta_j^{l+1} \tag{8}$$

The transition probability from state s_i to state s_j is:

$$\xi_{lij} = \frac{\alpha_i^l \beta_i^l}{\sum_{j=1}^{N} \alpha_j^l \beta_j^l} \tag{9}$$

M-step: After obtaining the transition probabilities between states, the parameters of MGHMM, including the component parameters and prior probabilities for each state, are re-evaluated.

The probability of the observation generated by the bth Gaussian of state s_i can be expressed as:

$$\gamma_{ilb} = \xi_{lij} \cdot \eta_{lb} = \frac{\alpha_i^l \beta_i^l}{\sum_{j=1}^{N} \alpha_j^l \beta_j^l} \cdot \frac{\omega_{ib} N(R; m, \Phi)}{\sum_{b=1}^{B} w_{ib} N(R; m, \Phi)} \tag{10}$$

The prior probability is:

$$\widehat{\varepsilon}_i = \frac{\sum_{b=1}^{B} w_{ib} N(R; m, \Phi)}{L} \tag{11}$$

The weight of the bth Gaussian distribution is:

$$\widehat{\omega}_{ilb} = \frac{\sum_{l=1}^{L} \gamma_{ilb}}{\sum_{l=1}^{L} \sum_{b=1}^{B} \gamma_{ilb}} \tag{12}$$

The mean of the bth Gaussian distribution can be given by:

$$\widehat{\mu}_{lib} = \frac{\sum_{l=1}^{L} \gamma_{ilb} \cdot r_l^n}{\sum_{l=1}^{L} \gamma_{ilb}} \tag{13}$$

The covariance of the bth Gaussian distribution is:

$$\eta_{lib} = \frac{\sum_{l=1}^{L} \gamma_{ilb} (r_l - \widehat{\mu}_{ib})(r_l - \widehat{\mu}_{ib})^T}{\sum_{l=1}^{L} \gamma_{ilb}} \tag{14}$$

Based on the derivation above, the state-based fingerprint of traffic f can be achieved. We can identify a target flow by calculating the occurrence probability of an observation sequence as follows:

$$\zeta_{ilb} = \wp(R|\lambda) = \sum_{l=1}^{L} \sum_{b=1}^{B} \alpha_i^l \beta_j^l \tag{15}$$

where $\wp(R|\lambda)$ donates the likelihood of the observation R under the condition λ, the training results of which allow us to choose a threshold θ. A flow from the testing traffic dataset will then be determined as traffic f when ζ is greater than θ; otherwise it is not the traffic f.

3.2. Training

The traffic hidden states classified manually has a big deviation. To obtain better classification results, we adopt the well-known K-means algorithm at the initial state classification. In order to obtain the best fitting effect, we choose different state clustering and Gaussian mixture components. For ease of understanding, the selected number of states in all the figures in this section is 3. For each observation state, we select the appropriate number of mixed Gaussians in the range of Dong and Jain (2019) and Aceto et al. (2019).

In Fig. 1, the x-axis is the logarithm of inter-packet time and the y-axis is the packet size. Fig. 1 shows the clustering result of K-means algorithm with 50 iterations when it has converged. The black cross in

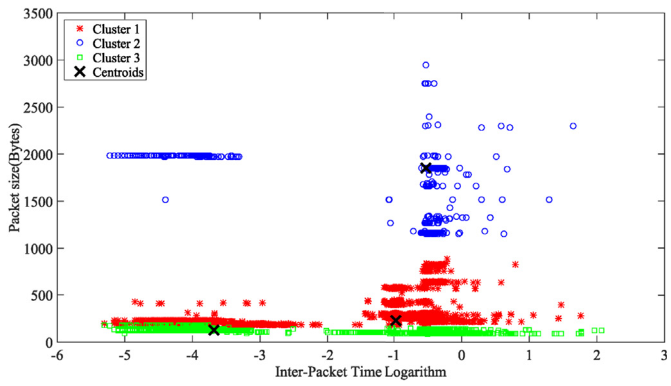


Fig. 1. Traffic classification with the K-means cluster algorithm.

Fig. 1 indicates the centroid position of each class. It can be seen from the cluster results that there is a clear boundary between the three types of states. Therefore, the cluster results can be taken as three different HMM states.

Based on cluster results obtained by the K-means algorithm, MGHMM trains to obtain the maximum likelihood of MOG parameters, i.e., weight, mean vector and covariance matrix of each cluster. To comprehensively evaluate how well MOG fits with each state cluster, the training results in this section will be presented in three parts: the first part shows how MOG fits with IPT; the second part shows how MOG fits with PS; the third part shows how MOG fits with the two-dimensional vector consisting of IPT and PS. It deserves attentions that IPT is usually small. However, in the case of network congestion or retransmission, the IPT is usually much larger than most IPTs. In light of its size, we use the logarithm of IPT sequence as both the training and testing inputs, to make sure that MOG fits well with IPT. On the other hand, the range of PS is narrowed as the ordinary packet size. Therefore, we only make use of the PS value.

Fig. 2 depicts the MOG IPT logarithm fitting results to traffic f after the training based on the three state clusters, where x-axis is the inter-packet time logarithm and y-axis is the occurrence of the inter-packet time. The blue bar is the number of current IPT logarithms in the training dataset, and the brown curve shows the fitting curve of the MOG to the IPT logarithms. Fig. 2(a)–(c) show the probability density distributions of the logarithm of IPT in the three states of HMM with MOG, respectively. Fig. 2(d) is the overall fitting result based on the fitting of the three HMM states. It can be seen from Fig. 2 that the peak and bottom of the density distribution in the four maps are basically the same as the mixed Gaussian distribution, and the peak values are basically the same. The curve of MOG can describe the strip distribution trend of IPT logarithm almost accurately. The reason is that the distribution of IPT is continuously changing in each state. It can be concluded that the MOG density function fits the IPT time logarithm very well.

After MGHMM training, the PS fitting results of the three types of states and all training datasets are shown in Fig. 3. This figure shows the PS fitting results of mixed Gaussian to traffic f based on the three state clusters, where x-axis denotes the packet size and y-axis is the probability of packet size. The blue bar is the number of current PSs in the training datasets, and the brown curve is the fitting curve of the MOG to the PSs. In the four plots, Fig. 3(a)–(c) are the probability density distributions of PSs fitted to the three states of HMM by MOG, and Fig. 3(d) is the overall fit of all PSs in the training dataset based on the three HMM state fits. It can be seen from Fig. 3 that the peak and bottom of the density distribution of training data in the four maps are basically the same as the MOG density distribution. The fitting effects of state 1 and state 3 are not as good as expected because the volume in these two states are relatively less in our traffic datasets. From the above four figures, it can be seen that the Gaussian density function can have a good fit of the PS.

The MOG fitting effect of the two-dimensional vector observations composed of IPT and PS, are evaluated in Fig. 4. In this figure, x-axis denotes the logarithm of inter-packet time and y-axis denotes packet size. Fig. 4 fits the overall variable density distribution formed by the two-dimensional vectors of IPT and PS. The results show that a Gaussian mixture can fit well for each type of observation. The place where the observation points are concentrated becomes the Gaussian center of the MOG, and each MOG can cover most of the observations. Those observations that cannot be covered need a larger training datasets. That is to say the training set used in this paper is not large enough, which is the reason why there are observations that are not covered by the MOG in Fig. 4(a).

After the above training, the proposed MGHMM becomes the maximum likelihood of traffic f with different states and Gaussian components. The traffic f identification is obtained based on the observation ranges characterized by those MGHMM parameters.

4. Experimental results and analysis

This section first introduces the evaluation indicators used in this paper, and then uses the self-captured traffic dataset and the public traffic dataset to evaluate the traffic classification performance of the MGHMM model.

4.1. Evaluation metrics

All traffic classification results can be classified into the following four results: True Positive (TP) represents the identification of the true traffic f as traffic f, and False Positive (FP) is the variable falsely identifying the ordinary traffic as traffic f. False Negative (FN) is the one falsely identifying traffic f as ordinary traffic, and True Negative (TN) is the one correctly identifying the true ordinary traffic as ordinary traffic. The evaluation indicators used in this paper to assess the performance of the proposed MGHMM are based on these four metrics, and can be expressed as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{19}$$

The above four evaluation indicators have been widely used in the related studies [29–34]. The *Accuracy* is identified as the proportion of correctly identified traffic, namely, the proportion of classified true traffic f from ordinary traffic with all tested traffic. The *Precision* is identified as the proportion of correctly classified traffic f, namely, the proportion of identified true traffic f with all the identified traffic f. *Recall* is the proportion of correctly identified traffic f with all the traffic f, namely, the proportion of the identified true traffic f with all the true traffic f. F_1 is an important indicator to evaluate the effectiveness of the classification method.

4.2. Experimental results

This section mainly includes three aspects, 1) verifying the effectiveness of traffic classification from two perspectives, i.e., protocol-level and application-level; 2) comparing our MGHMM model with the latest VQ-based traffic classification algorithm; 3) optimizing the model parameters selection of MGHMM.

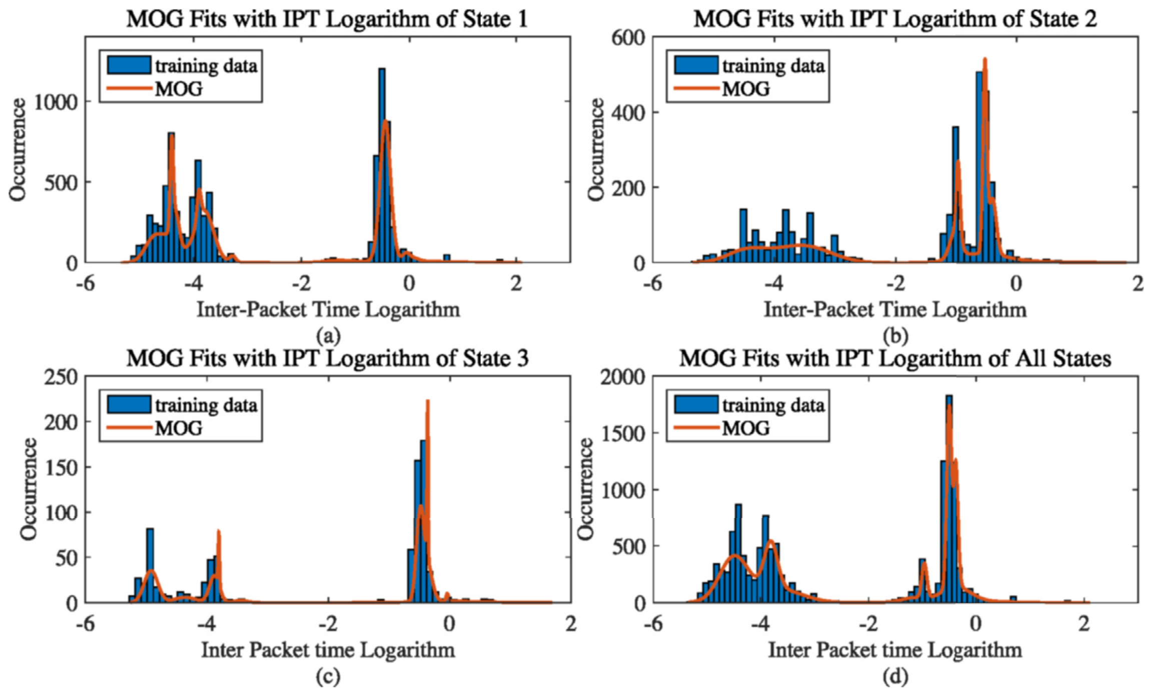


Fig. 2. MGHMM portrays the hidden states of IPTs logarithm. The sub-figures (a)–(c) show the fitness of IPT logarithm with MOG in states 1–3, respectively, and (d) shows the fitness of IPT logarithm with MOG in all states of IPTs.

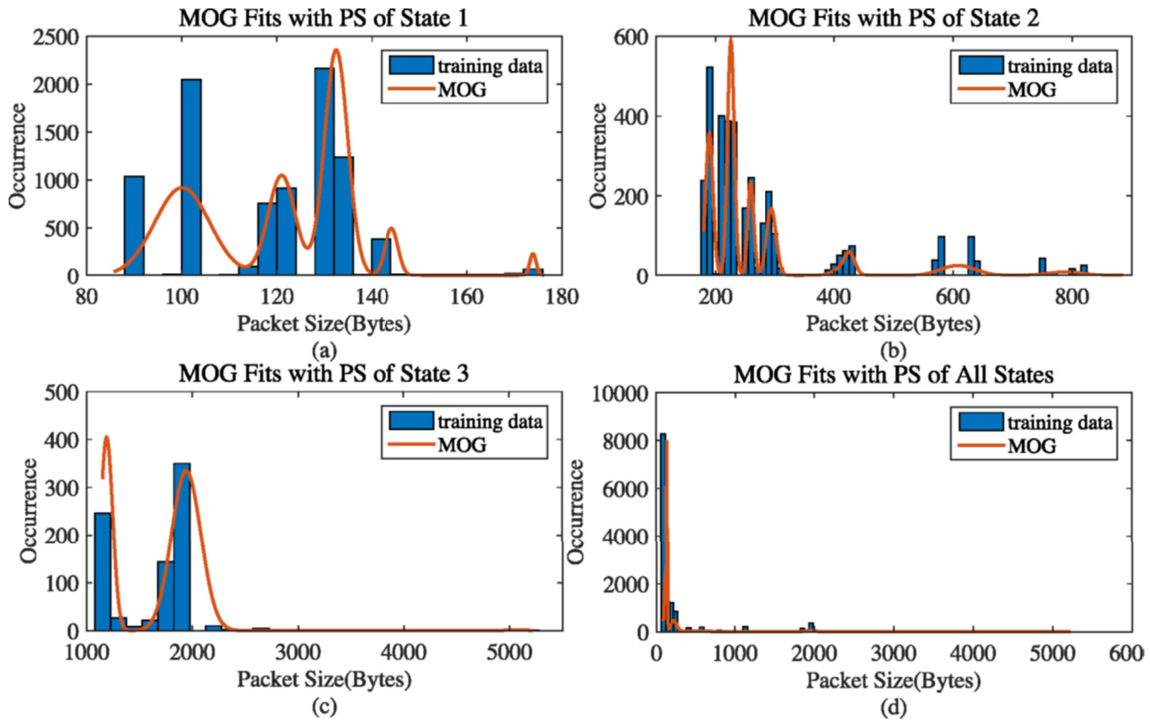


Fig. 3. MGHMM depicts the hidden states of PSs. The sub-figures (a)–(c) present the fitness of PS with MOG in states 1–3, respectively, and (d) shows the fitness of PS with MOG in all state PSs.

4.2.1. *Traffic classification*

To verify the validity and effectiveness of the proposed MGHMM, this paper first verifies the model's performance in protocol-level (i.e. HTTP, SMB, STUN, TLS, etc.) traffic classification with the UNB open traffic dataset (Sharafaldin et al., 2018) and in application-level (i.e., meek, FTE, Obfs, etc.) obfuscation traffic classification with the traffic dataset intercepted in the lab environment.

- Classification based on protocol-level traffic

We use the Wireshark tool to mark the protocol of the public dataset and extract the binary feature vector (IPT, PS). The main protocol types included in the traffic dataset and the classification results of MGHMM are shown in Fig. 5. In this figure, x-axis indicates different types of protocols and y-axis is the value of *Precision*, *Accuracy*, *Recall*, F_1 of those protocols. Among those protocols, FTP (File Transfer Protocol), TLSv1

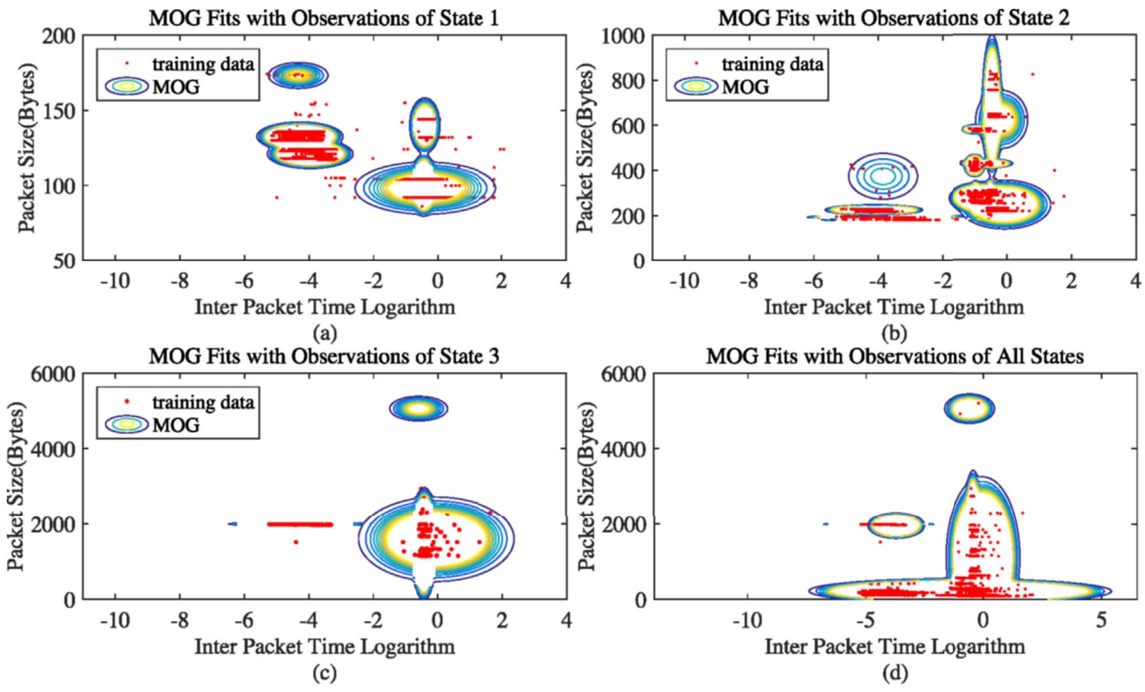


Fig. 4. MGHMM shows the hidden states of observations. The sub-figures (a)–(c) show the fitness of the two-tuple observation consisting of IPT logarithm and PS with MOG in states 1–3, respectively, and (d) depicts the fitness of the two-tuple observation consisting of IPT logarithm and PS with MOG in all states for the two-tuples.

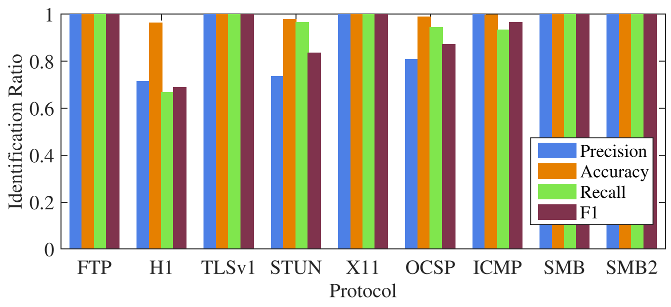


Fig. 5. Traffic classification for protocol-level traffic.

(Transport Layer Security version 1), X11 (X Window System ProtocolX Version 11), SMB (Server Message Block) and SMB2 (Server Message Block version 2) can be classified clearly with 99% of all four indicators, while the metrics of H1 (SINEC H1 protocol), STUN (Session Traversal Utilities for NAT) and OCSP (the Online Certificate Status Protocol) are lower relatively. H1 is a simple protocol to transfer PLC (programmable logic controller) data between Siemens SIMATIC S5 devices. STUN is a standardized set of methods for traversal of network address translator (NAT) gateways in applications of real-time voice, video, messaging and other interactive communications. H1 and STUN are used for real-time large data transition. Therefore, IPT and PS of H1 and STUN are relatively more concentrated, which is why they are more difficult to classify. OCSP is used for obtaining the revocation status of an X.509 digital certificate, which is usually in the front of HTTP or other application layer protocol. That is the reason why the evaluation metrics are lower.

- Classification based on application-level obfuscation traffic

To evaluate the effectiveness and accuracy of the proposed MGHMM in the application-level, we use our model to identify the Meek-based Tor traffic, which is an obfuscated application traffic. The traffic dataset is 15 GB in size, including web pages, emails, images, videos, audios, and files (e.g., executables, source code, and PDF files), generated by browsers such as Chrome, Firefox, Tor Browser, Arora, etc. All the traffic is captured with a browser driver script on the gateway of a local area network.

Based on the training results in Section 3.2, we make the corresponding experiment of Meek-based Tor traffic identification. The evaluation results are shown in Fig. 6, where x-axis is the number of Gaussian components of each state and y-axis denotes the traffic identification accuracy. The results are under the conditions that the traffic observations are divided into 3 states, 6 states and 10 states, and each state consists of 7–15 Gaussian components. From Fig. 6, we can see that the more states are divided, the higher classification accuracy is, if each state is characterized with the same number of Gaussians. Similarly, the larger the number of Gaussians is, the higher classification accuracy is, when the traffic has the same states volume. It is worth noting that the evaluation metrics tend to be less steady when fewer states are divided. The main reason is that some observations far away from a state cluster center is forcedly clustered into the state when there are only a few clusters. When the test traffic is divided into 10 states and each state has 11 Gaussian components, the Tor traffic identification results can be seen from Fig. 6, that the proposed MGHMM can get 99.98% on accuracy, 100% on precision, 99.43% on recall, and 99.72% on F_1. The reason why such good results can be obtained is that, on the one hand, the adoption of K-means algorithm provides a good initial classification result, and on the other hand, MOG is used to characterize the observed density distribution of each state which can cover almost each peak and trough. The experiment results indicate that MGHMM can effectively identify the Meek-based Tor traffic.

4.2.2. *Comparison of MGHMM with the latest VQ-based algorithm*

Tree Structured Vector Quantise (TSVQ) uses a tree-based search method to improve the VQ algorithm, and the data in each branch of the tree is clustered. The Parallel Tree Structured Vector Quantiser (PTSVQ) algorithm is an extension of the tree structure vectorizer, attempting to represent input data using multiple codes. PTSVQ creates a TSVQ instance for each class, with each data point running through all TSVQ

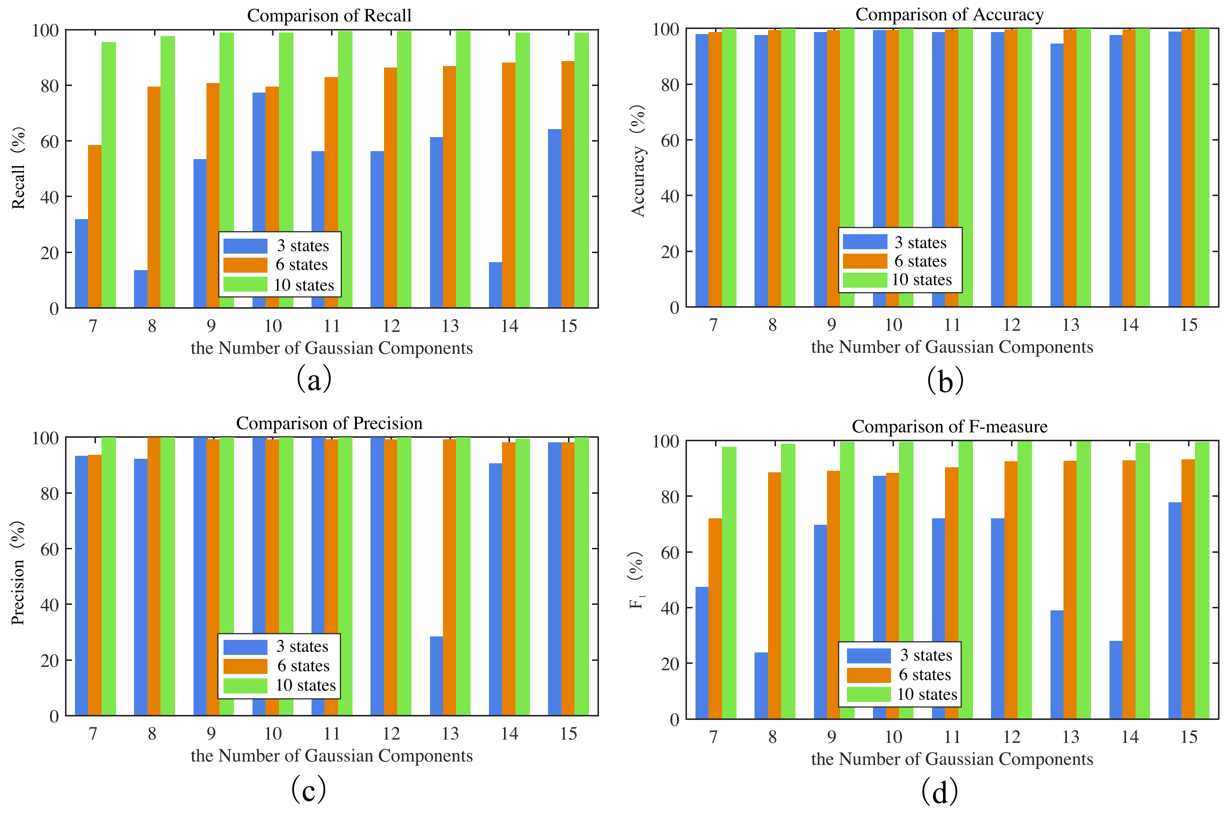


Fig. 6. Traffic classification comparison with different parameters. The sub-figures (a)–(d) show the comparison results with different states (i.e., 3 states, 6 states and 10 states) for *Recall*, *Accuracy*, *Precision* and F_1, respectively.

instances and being assigned to the class that produces the smallest distortion metric for the model.

Jeo et al. (Kornycky et al., 2017) proposed the TRee Adaptive Parallel Vector Quantiser (TRAP-VQ) using the well-known vector quantisation algorithm in conjunction with a decision tree. TRAP-VQ performs vector quantisation preprocessing on the public dataset, constructs a new VQ encoding function, and builds a tree structure. The branch tags are *target*, *non-target*, and *continue*. As the latest traffic classification research work, VQ-based model is simple in design and has very high practical value. Besides, its experimental results in Kornycky et al. (2017) show that TRAP-VQ and PTSVQ have good effects in traffic classification. Thus, we use TRAP-VQ and PTS-VQ for classification performance comparison to show merits of our proposed scheme. To illustrate the effectiveness of our model, this paper uses the traffic used in Wang et al. (2015) to compare MGHMM with the latest VQ-based traffic classification algorithms, and selects two traffic recognition scenarios, i.e., protocol identification and obfuscated traffic identification.

To implement the protocol identification comparison, we choose four commonly used protocols, i.e., FTP, TLSv1, SSH and HTTP from the dataset. It can be seen from Fig. 7 that MGHMM outperforms TRAP-VQ and PTS-VQ on the four types of traffic protocols. From Recall, Accuracy and Precision, it can be clearly seen that the MGHMM classification performance is better than TRAP-VQ's, and TRAP-VQ's performance is better than PTS-VQ. For protocol FTP, TLSv1 and SSH, MGHMM's performance is 2% higher than TRAP-VQ and PTS-VQ in terms of Accuracy. In addition, MGHMM has a recall rate that is nearly 20% higher than that of TRAP-VQ and PTS-VQ in identifying TLSv1. MGHMM has a recall that is not less than that of TRAP-VQ and PTS-VQ in identifying FTP and SSH. MGHMM's performance is 2% higher than TRAP-VQ and PTS-VQ in terms of Precision and in F_1. Regrettably, the classification performance of MGHMM is lower than TRAP-VQ when identifying HTTP traffic. The reason is that HTTP flows are very short and their time intervals are very small, while our MGHMM model is more effective for relatively long flows.

The identification of obfuscated traffic is the original purpose of MGHMM. To highlight the advantages of MGHMM in this regard, we also use PTS-VQ and TRAP-VQ identification methods to compare with MGHMM. From Fig. 8, it is clear to see that MGHMM has very obvious advantages over PTS-VQ and TRAP-VQ in identifying obfuscated traffic. The recognition effect of other obfuscated traffic is higher than PTS-VQ and TRAP-VQ algorithms. The most prominent is that the accuracy of MGHMM is 10% higher than the PTS-VQ and TRAP-VQ algorithms in identifying Google's meek obfuscated traffic. The reason why such a good effect can be obtained is mainly because the confused traffic, such as Meek and Obfs, have adopted methods like transmission interval interference and message filling. While, the Format-Transforming Encryption (FTE) obfuscation method uses the regularization method to shape the traffic content, which has a small impact on the packet interval and packet size. This is why the recognition effect of FTE is lower than the recognition rate of VQ-based algorithms.

4.2.3. *The trade-off between classification and computational cost*

It is not a good practice to use many Gaussian components. For example, the application-level classification in Section 4.2.1 shows that the accuracy with 14 and 15 Gaussians are lower than that with 11 Gaussians. In addition, the model with 14 and 15 Gaussians take longer calculation time. In the classification process, a small number of state divisions and Gaussian components can cause the accuracy to decrease. However, a large number of state divisions and Gaussian components mixed will bring higher computational overhead when the accuracy reaches a certain value, and the accuracy will not be improved further. With an appropriate number of state divisions and Gaussian components, not only can the computational overhead be reduced but also an optimal classification performance can be obtained. To achieve the better ratio between classification and cost, this paper uses F_1 and pro-

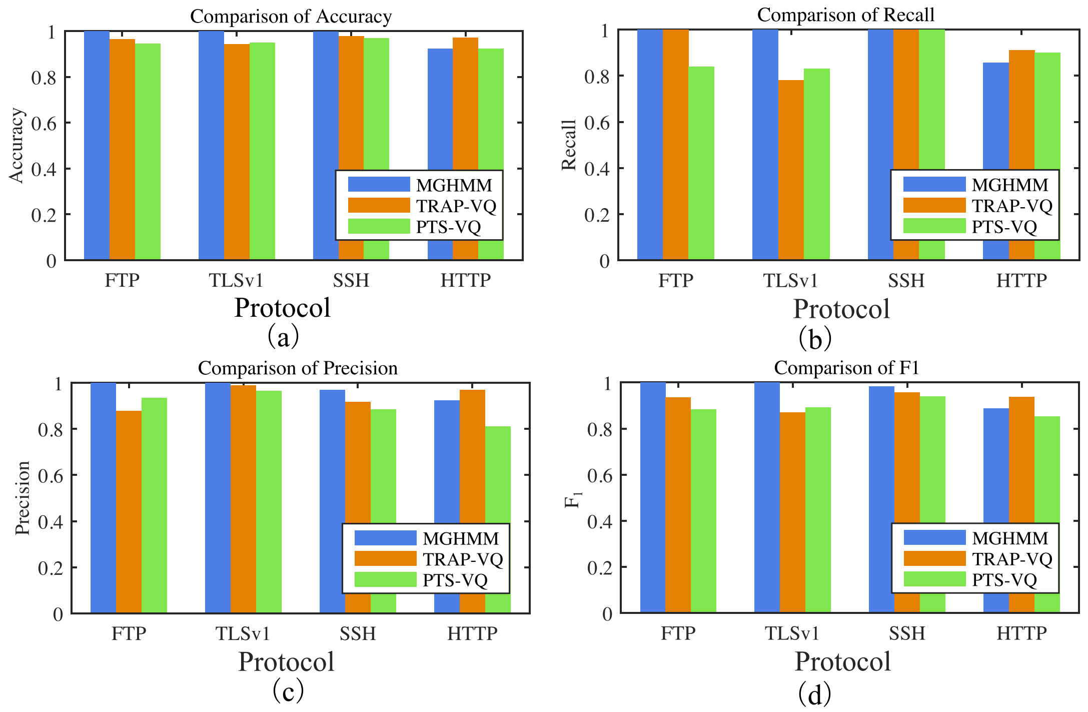


Fig. 7. Traffic classification comparison with different methods. The sub-figures (a)–(d) depict the comparison results among MGHMM, TRAP-VQ and PTS-VQ for classifying protocol-level traffic (i.e., FTP, TLSv1, SSH and HTTP), for *Accuracy*, *Recall*, *Precision*, and F_1, respectively.

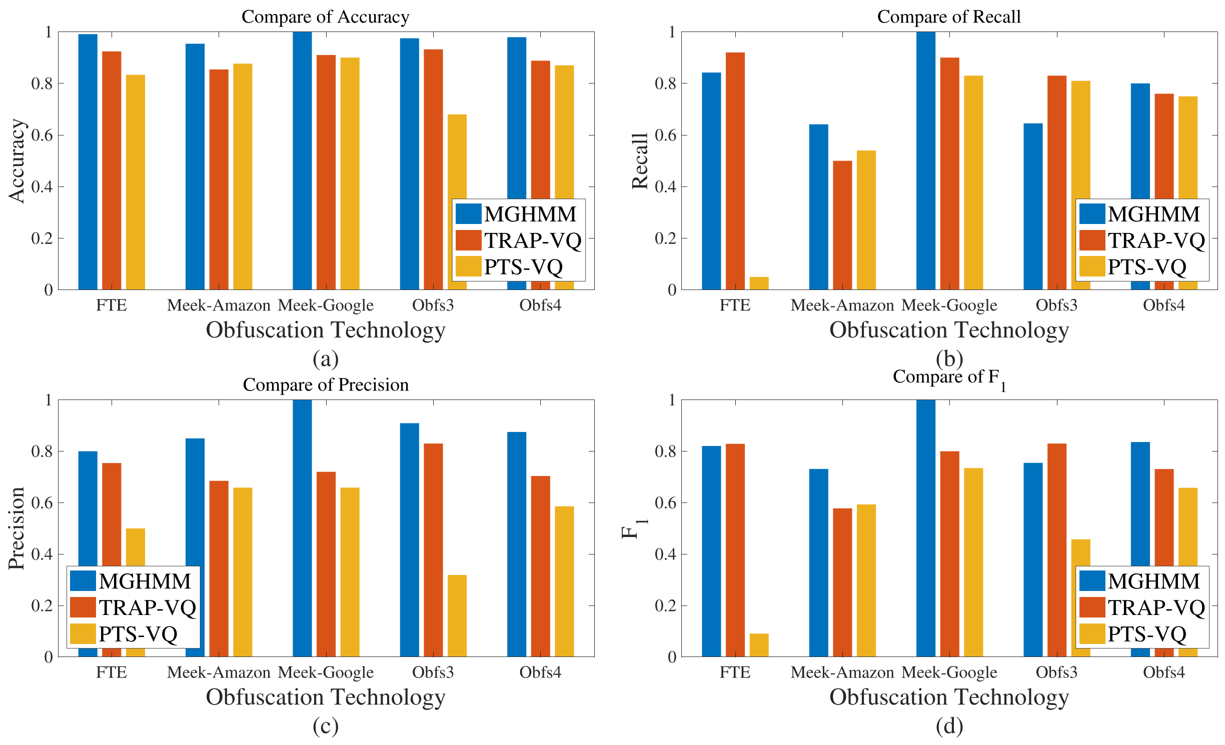


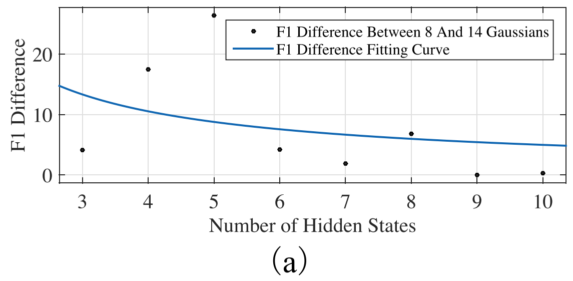
Fig. 8. Obfuscation traffic classification comparisons with different methods. The sub-figures (a)–(d) show the comparison results among MGHMM, TRAP-VQ and PTS-VQ for classifying obfuscated traffic (i.e., FTE, Meek-Amazon, Meek-Google, Obfs3 and Obfs4), for *Accuracy*, *Recall*, *Precision* and F_1, respectively.

cessing time to optimize our model. F_1 is a comprehensive indicator calculated by *Precision* and *Recall*, and longer processing time represents larger cost within the same physical conditions.

- F_1 increment regression

It can be seen from Fig. 9(a) that the number of hidden states of the depicted traffic and the number of mixed Gaussians affect the classification performance. To this end, we explore the relationship between F_1 and the number of hidden states and mixed Gaussians.

Firstly, the number of hidden states is used as a variable. The increment Ic_f^s of F_1 between the mixed 8 Gaussians and the mixed 14 Gaus-



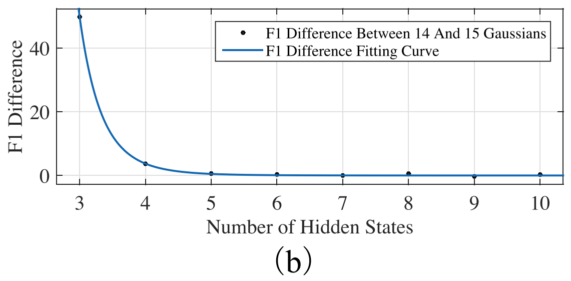


Fig. 9. F_1 difference comparison between different Gaussians: (a) F_1 difference between 8 and 14 Gaussians and its fitting curve, and (b) F_1 difference between 14 and 15 Gaussians and its fitting curve.

sians is calculated. The increment Ic_f^s of F_1 between 14 Gaussians and 15 Gaussians is mixed. From Fig. 9 (a), it can be seen that there is a tendency of decreasing the F_1 increment Ic_f^s as the number of hidden states increase. This is also shown by the blue trend curve. This trend curve, subject to a power function, can be formalized with the parameters $\{a_s, b_s, c_s\}$ as follows:

$$Ic_f^s = Fscore_s(s) = a_s s^{b_s} + c_s \tag{20}$$

where Ic_f^s is the increment of F_1 affected by the number of hidden states. It can be seen from Fig. 9(b) that the F_1 increment between 14 and 15 mixed Gaussians is also a decreasing trend corresponds to a power function with a negative exponent, as the fitted blue trend curve. Although this curve is decreasing faster, it can still be described by Eq. (20). The only difference is that the value of parameters of the fitting function is different. The parameter b_s in Fig. 9(b) is smaller.

Secondly, with the number of mixed Gaussians increasing as the variable, the F_1 increment Ic_f^g between 6 and 8 hidden states and the same values between 8 and 10 states are calculated. From Fig. 10(a), it can be seen that Ic_f^g has a decreasing trend corresponding to a power function with a negative exponent as the number of mixed Gaussians increases. The blue trend curve can also be formalized with parameters $\{a_g, b_g, c_g\}$ as follows:

$$Ic_f^g = Fscore_g(g) = a_g g^{b_g} + c_g \tag{21}$$

where Ic_f^g is the increment of F_1 affected by the number of mixed Gaussians. It can be seen from Fig. 10(a) that there is a decreasing tendency of the F_1 increments as the increasing of Gaussians number between the clustered 6 hidden states and 8 hidden states observations, whose fitted curve is a power function with a negative exponent as shown in Fig. 9(a) and (b). That is to say, Fig. 10(a) can still be described by Eq. (21). Fig. 10(b) shows the same decreasing tendency of the F_1 increments between the clustered 8 hidden states and 10 hidden states observations as depicted in Fig. 10(a). The differences between Fig. 10(a) and (b) are the different parameters of the fitting function, such as the parameter b_g in Fig. 10(b) is smaller.

The number of hidden states and the number of mixed Gaussian distributions are constructed as binary variable vectors **x** = [s, g]. According to Eqs. (20) and (21), an F_1 function that varies with the vector **x** is formed:

$$\begin{aligned} Ic_f &= Fscore(\mathbf{x}) = Fscore(s,g) \\ &= \delta_f Fscore_s(s) + \varrho_f Fscore_g(g) \\ &= \delta_f(a_s s^{b_s} + c_s) + \varrho_f(a_g g^{b_g} + c_g) \end{aligned} \tag{22}$$

where Ic_f is the increment of F_1, which is affected by the number of hidden states and the mixed Gaussian. δ_f, ϱ_f are used to adjust the degree of influence of the hidden state and the mixed Gaussian on F_1.

- Cost increment regression

The number of different hidden states and the number of mixed Gaussians not only affect the performance of classification, but also cause the difference of computational overhead of the MGHMM model. Under the same experimental data and physical resources, we explore the relation between the computation cost and the number of hidden states and mixed Gaussians.

Firstly, the number of hidden states is used as a variable to calculate the cost increment Ic_c^s between the mixing of 8 Gaussians and the mixing of 14 Gaussians, and the cost increment Ic_c^s between 14 Gaussians and 15 Gaussians is mixed. From Fig. 11(a), it can be seen that there is a tendency of increasing the cost increment of the mixed 8 Gaussians and 14 Gaussians. The trend curve, subject to a linear function, can be formalized with the parameters $\{m_s, n_s\}$ as follows:

$$Ic_c^s = Rcost_s(s) = m_s s + n_s \tag{23}$$

where Ic_c^s is the increment of computational cost affected by the number of hidden states. It can be seen from Fig. 11(b) that there is also a tendency to increase the cost increments of the mixed 14 Gaussians and 15 Gaussians. This curve is almost the same with Eq. (23). The parameters $\{m_s, n_s\}$ should be adjusted slightly because its slope and interception is different. As can be seen from in Fig. 11(a) and (b), the more the number of hidden states is divided, the faster the computation overhead increases.

Secondly, with the number of mixed Gaussians increasing as a variable, the cost increment Ic_c^g between 6 and 8 clustered states and the same value between 8 and 10 clustered states are calculated, which show a linear increase. From Fig. 12(a), the linear increase tendency is fitted with a blue line. The blue trend line can be formalized with parameters $\{m_g, n_g\}$ as follows:

$$Ic_c^g = Rcost_g(g) = m_g g + n_g \tag{24}$$

where Ic_c^g is the increment of computational cost affected by the number of mixed Gaussians. It can be seen from Fig. 12(b) that there is also a tendency of linear increase of the cost increments between 8 and 10 clustered states with the increase of the number of mixed Gaussians. When we fit the cost increment, the blue trend line shows that it can continue to describe its trend using Eq. (24).

The number of hidden states and the number of mixed Gaussians are constructed as binary variable vectors **x** = [s, g]. According to Eqs. (23) and (24), the cost function varies with the vector **x**:

$$\begin{aligned} Ic_c &= Rcost(\mathbf{x}) = Rcost(s,g) \\ &= \delta_c Rcost_s(s) + \varrho_c Rcost_g(g) \\ &= \delta_c(m_s s + n_s) + \varrho_c(m_g g + n_g) \end{aligned} \tag{25}$$

where Ic_c is the increment of computational cost, which is affected by the number of hidden states and the mixed Gaussians. δ_c, ϱ_c are used to adjust the degree of influence of the hidden state and the mixed Gaussians on the computational cost.

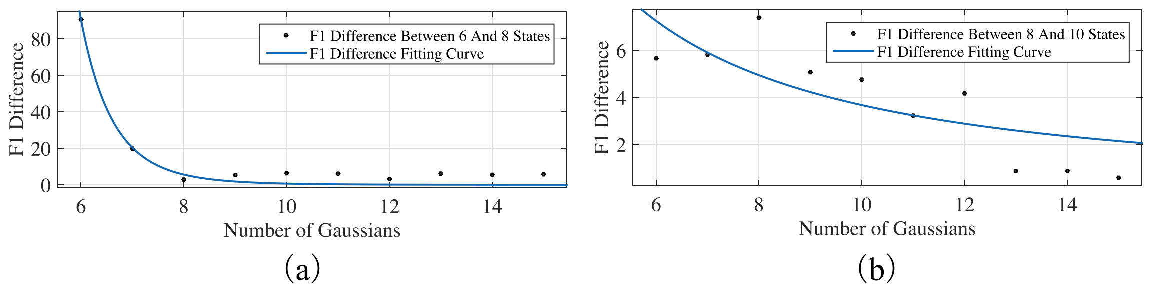


Fig. 10. F_1 difference comparison between different hidden states: (a) F_1 difference between 6 and 8 states and its fitting curve, and (b) F_1 difference between 8 and 14 states and its fitting curve.

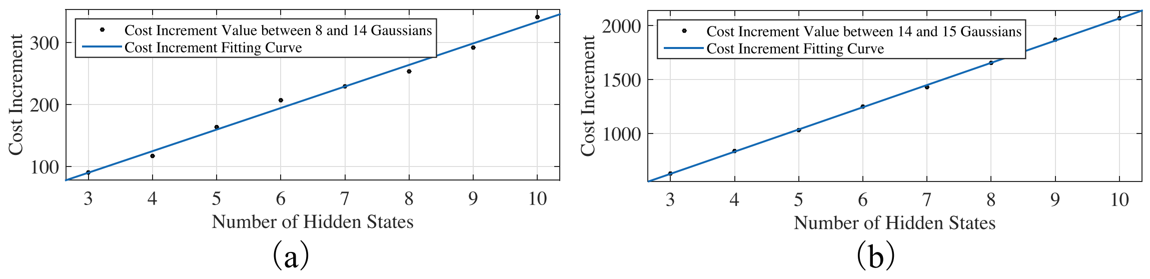


Fig. 11. Cost increment comparison between different Gaussians in testing phase: (a) Cost increment between 8 and 14 Gaussians as the number of states increasing and its fitting curve, and (b) Cost increment between 14 and 15 Gaussians as the number of states increasing and its fitting curve.

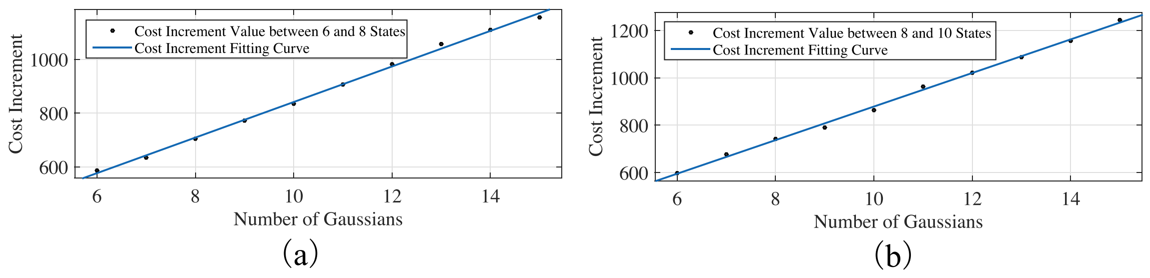


Fig. 12. Cost increment comparison between different hidden states in testing phase: (a) Cost increment between 6 and 8 states as the number of Gaussians increasing and its fitting curve, and (b) Cost increment between 8 and 10 states as the number of Gaussians increasing and its fitting curve.

4.2.4. State and Gaussian component decisions

After the analysis in Section 4.2.3, it can be found that the computational cost is gradually increased and F_1 increases with the number of hidden states and the number of mixed Gaussians until reaching a split point $[s_t, g_t]$. It is unwise to simply chase the upgrade of F_1 and ignore the increasing cost. Our goal is to search for the best value $[s_{op}, g_{op}]$ from all the possible hidden states and mixed Gaussians combination set.

We know that both F_1 and cost increase with the number of hidden states and the number of mixed Gaussians, but the growth rate of F_1 and cost is different. The variations of F_1 and computational cost have been expressed geometrically. To determine whether the state quantity selection and the Gaussian quantity selection are optimal for a certain type of traffic, we define the growth function as shown below:

$$K(\mathbf{x}) = Fscore(\mathbf{x}) - \rho Rcost(\mathbf{x}) \tag{26}$$

The growth function is not what we ultimately need. Our aim is that the construction can be a balance between the F_1 growth rate and the calculation overhead growth rate, so Eq. (27) is derived as follows, which is called discriminant function.

$$\begin{aligned}\kappa(\mathbf{x}) &= \frac{\partial K(\mathbf{x})}{\partial \mathbf{x}} \\ &= \frac{\partial Fscore(\mathbf{x})}{\partial \mathbf{x}} - \rho\frac{\partial Rcost(\mathbf{x})}{\partial \mathbf{x}}\end{aligned} \tag{27}$$

where ρ is parameter used for balancing the two slopes of both classification and computational cost increment function. When $\kappa(\mathbf{x}) = 0$, the value of $\mathbf{x} = [s_i, g_i]$ at this time is the optimal model parameter. The growth rate of classification result and computation cost of MGHMM are two important factors to determine the parameter selection and are mainly affected by the number of hidden states and mixed Gaussians.

To choose the best number of hidden states, we first find the partial derivative of Eq. (28) as follows:

$$\begin{aligned}\kappa(\mathbf{x})_s = \kappa(s,g)_g &= \frac{\partial K(s,g)}{\partial s} \\ &= \frac{\partial Fscore(s,g)}{\partial s} - \rho\frac{\partial Rcost(s,g)}{\partial s} \\ &= \frac{\partial(\delta_f(a_s s^{b_s} + c_s) + \varrho_f(a_g g^{b_g} + c_g))}{\partial s} \\ &\quad - \rho\frac{\partial(\delta_c(m_s s + n_s) + \varrho_c(m_g g + n_g))}{\partial s} \\ &= \delta_f a_s b_s s^{b_s - 1} - \rho\delta_c m_s\end{aligned} \tag{28}$$

To choose the best number of mixed Gaussians, we find the partial derivative of Eq. (29) as follows:

$$\begin{aligned}\kappa(\mathbf{x})_g &= \kappa(s,g)_g = \frac{\partial K(s,g)}{\partial g} \\ &= \frac{\partial Fscore(s,g)}{\partial g} - \rho\frac{\partial Rcost(s,g)}{\partial g} \\ &= \frac{\partial(\delta_f(a_s s^{b_s}+c_s)+\varrho_f(a_g g^{b_g}+c_g))}{\partial g} \\ &\quad - \rho\frac{\partial(\delta_c(m_s s+n_s)+\varrho_c(m_g g+n_g))}{\partial g} \\ &= \varrho_f a_g b_g s^{b_g-1} - \rho\varrho_c m_g \end{aligned} \tag{29}$$

To verify Eqs. (28) and (29), we use the application-level classification experiment in Section 4.2.1 for verification. The results of Meek-based Tor experiments show that the optimal parameters of Meek-based Tor flow are 10 hidden states, and each hidden state adopts 12 Gaussians (after rounding), which is consistent with the classification results shown in Fig. 6.

5. Conclusions

This paper has proposed a traffic classification model MGHMM, which is based on hidden Markov model. MGHMM uses K-means clustering algorithm, hidden Markov model and mixed Gaussians to gradually reduce the flow of details. To evaluate and analyze the traffic classification effect of our model, firstly, this paper has used a public traffic dataset and a private traffic dataset collected from a self-built LAN, to evaluate the traffic classification in protocol-level and application-level traffic. The experiments have shown a good result on the performance of the proposed scheme. Secondly, we have compared MGHMM with the latest VQ-based traffic classification algorithms and have found that the proposed MGHMM has obvious advantages in traffic classification. Thirdly, a new model parameter selection method has been proposed, which can achieve better classification results with lower computational cost.

CRediT authorship contribution statement

Zhongjiang Yao: Methodology, Data curation, Writing - original draft, Writing - review & editing. **Jingguo Ge:** Conceptualization, Methodology. **Yulei Wu:** Formal analysis, Writing - review & editing. **Xiaosheng Lin:** Supervision, Visualization. **Runkang He:** Software, Validation. **Yuxiang Ma:** Investigation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the National Science and Technology Major Project of the Ministry of Science and Technology of China (Grant No.: 2017ZX03001019-003), the National Key R&D Plan "Cyberspace Security" Major Projects of China (Grant No.: 2017YFB0801801), the Science and Technology Service Network Initiative (STS) Project of Chinese Academy of Science (Grant No.: Y7X0071105), the Young Elite Scientist Sponsorship Program by Henan Association for Science and Technology (No. 2020HYTP008), and the Key Scientific and Technological Project of Henan Province (No. 202102210352).

References

Aceto, G., Ciuonzo, D., Montieri, A., Pescap, A., 2018. Multi-classification approaches for classifying mobile app traffic. J. Netw. Comput. Appl. 103, 131–145. https://academic.microsoft.com/paper/2768726319.
Aceto, G., Ciuonzo, D., Montieri, A., Pescap, A., 2019. Mimetic: mobile encrypted traffic classification using multimodal deep learning. Comput. Network. 165, 106944. https://academic.microsoft.com/paper/2981318525.
Auld, T., Moore, A.W., Gull, S.F., 2007. Bayesian neural networks for internet traffic classification. IEEE Trans. Neural Network. 18 (1), 223–239.
Casino, F., Choo, K.-K.R., Patsakis, C., 2019. Hedge: efficient traffic classification of encrypted and compressed packets. IEEE Trans. Inf. Forensics Secur. 14 (11), 2916–2926. https://academic.microsoft.com/paper/2953225812.
Chen, Z., Wen, J., Geng, Y., 2016. Predicting future traffic using hidden markov models. In: 2016 IEEE 24th International Conference on Network Protocols (ICNP), pp. 1–6.
Dainotti, A., de Donato, W., Pescape, A., Rossi, P.S., 2008. Classification of network traffic via packet-level hidden markov models. In: IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, pp. 1–5.
Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the em algorithm. J. Roy. Stat. Soc. Ser. b-Methodol. 39 (1), 1–38.
Desai, D., 2018. February 2018 Zscaler Ssl Threat Report. https://www.zscaler.com/blogs/research/february-2018-zscaler-ssl-threat-report.
Dingledine, R., Mathewson, N., Syverson, P.F., 2004. Tor: the second-generation onion router. In: SSYM04 Proceedings of the 13th Conference on USENIX Security Symposium, vol. 13, 2121.
Dong, S., Jain, R., 2019. Flow online identification method for the encrypted skype. J. Netw. Comput. Appl. 132, 75–85. https://academic.microsoft.com/paper/2914899172.
Fan, J.L., 2001. Forward-backward Algorithm, vol. 627. Springer International, pp. 97–116.
Georgoulas, S., Moessner, K.e. a., 2010. Towards efficient protocol design through protocol profiling and verification of performance and operational metrics. In: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference, IWCMC10. ACM, New York, NY, USA, pp. 848–852, https://doi.org/10.1145/1815396.1815591.
Huang, C., Min, G., Wu, Y., Ying, Y., Pei, K., Xiang, Z., 2017. Time series anomaly detection for trustworthy services in cloud computing systems. IEEE Trans. Big Data, https://doi.org/10.1109/TBDATA.2017.2711039 11.
Karagiannis, T., Broido, A., Brownlee, N., Claffy, K.C., Faloutsos, M., 2004. Is p2p dying or just hiding? [p2p traffic measurement]. In: IEEE Global Telecommunications Conference, 2004. GLOBECOM 04, vol. 3, pp. 1532–1538.
Karagiannis, T., Papagiannaki, K., Faloutsos, M., 2005. Blinc: multilevel traffic classification in the dark. ACM Spec. Interest Group Data Communication 35 (4), 229–240.
Korczynski, M., Duda, A., 2014. Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pp. 781–789.
Kornycky, J., Abdul-Hameed, O., Kondoz, A.M., Barber, B.C., 2017. Radio frequency traffic classification over wlan. IEEE/ACM Trans. Netw. 25 (1), 56–68.
Liu, Z., Wang, R., Japkowicz, N., Cai, Y., Tang, D., Cai, X., 2019a. Mobile app traffic flow feature extraction and selection for improving classification robustness. J. Netw. Comput. Appl. 125, 190–208. https://academic.microsoft.com/paper/2898307979.
Liu, C., He, L., Xiong, G., Cao, Z., Li, Z., 2019b. Fs-net: a flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 1171–1179. https://academic.microsoft.com/paper/2919493784.
Pan, W., Cheng, G., Tang, Y., 2017. Wenc: Https encrypted traffic classification using weighted ensemble learning and Markov chain. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 50–57.
Prince, M., 2016. The Trouble with Tor. https://blog.cloudflare.com/the-trouble-with-tor/.
Queiroz, W., Capretz, M.A.M., Dantas, M.A.R., 2019. An approach for sdn traffic monitoring based on big data techniques. J. Netw. Comput. Appl. 131, 28–39. https://academic.microsoft.com/paper/2914593650.
Rezaei, S., Liu, X., How to achieve high classification accuracy with just a few labels: a semi-supervised approach using sampled packets. arXiv preprint arXiv:1812.09761 https://academic.microsoft.com/paper/2906493541.
Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, pp. 108–116. https://academic.microsoft.com/paper/2789828921.
Tahaei, H., Afifi, F., Asemi, A., Zaki, F., Anuar, N.B., 2020. The rise of traffic classification in iot networks: a survey. J. Netw. Comput. Appl. 154, 102538.
Tan, X., Xie, Y., Ma, H., Yu, S., Hu, J., 2019. Recognizing the content types of network traffic based on a hybrid dnn-hmm model. J. Netw. Comput. Appl. 142, 51–62.
Wang, L., Dyer, K.P., Akella, A., Ristenpart, T., Shrimpton, T., 2015. Seeing through network-protocol obfuscation. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 57–69. https://academic.microsoft.com/paper/2052565897.
Wang, P., Chen, X., Ye, F., Sun, Z., 2019. A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access. 7, 54024–54033. https://academic.microsoft.com/paper/2942239200.
Wright, C.V., Monrose, F., Masson, G.M., 2004. Hmm profiles for network traffic classification. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 9–15.
Yao, Z., Ge, J., Wu, Y., Zhang, X., Li, Q., Zhang, L., Zou, Z., 2018. Meek-based tor traffic identification with hidden markov model. 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. 335–340. https://academic.microsoft.com/paper/2913669552.

Zou, Z., Ge, J., Zheng, H., Wu, Y., Han, C., Yao, Z., 2018. 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. https://academic.microsoft.com/paper/2911301616.

Zhongjiang Yao received a Ph.D. degree in computer system architecture from Institute of Information Engineering, Chinese Academy of Sciences in 2019. His current research interests include network security, blockchain and machine learning.

Jingguo Ge received a Ph.D. degree in computer system architecture from the Institute of Computing Technology of the Chinese Academy of Sciences in 2003. He is currently a professor at Institute of Information Engineering, Chinese Academy of Sciences, and a professor at School of Cyber Security, University of Chinese Academy of Sciences. His research focuses on computer network architecture, software defined network (SDN), network virtualization, cyber security, and mobile communication networks.

Yulei Wu is a Senior Lecturer with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, United Kingdom. He received the B.Sc. degree (1st Class Hons.) in Computer Science and the Ph.D. degree in Computing and Mathematics from the University of Bradford, United Kingdom, in 2006 and 2010, respectively. His expertise is on networking and his main research interests include computer networks, networked systems, software defined networks and systems, network management, and network security and privacy. He is an Editor of IEEE Transactions on Network and Service Management, Computer Networks (Elsevier) and IEEE Access. He contributes to major conferences on networking as various roles including the Steering Committee Chair, the General Chair and the Program Chair. His research has been supported by Engineering and Physical Sciences Research Council of United Kingdom, National Natural Science Foundation of China, University's Innovation Platform and industry. He is a Senior Member of the IEEE, and a Fellow of the HEA (Higher Education Academy).

Xiaosheng Lin received a Ph.D. degree from Jilin University in 2017. She is currently an associate professor at Jilin Province Academy of Environmental Sciences.

Runkang He is a master student in computer system architecture at the Institute of Information Engineering, Chinese Academy of Sciences. His current research interests include network anomaly detection and natural language processing.

Yuxiang Ma is currently an associate professor in the School of Computer and Information Engineering at Henan University. He received the B.S. degree from Henan University in 2013, and the Ph.D. degree from the Computer Network Information Center, Chinese Academy of Sciences in 2019. His main research interests include future Internet architecture and technologies, network security, and privacy enhancement technologies.