

Received March 18, 2020, accepted April 10, 2020, date of publication May 6, 2020, date of current version May 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992556

# Timely Classification and Verification of Network Traffic Using Gaussian Mixture Models

HASSAN ALIZADEH<sup>1</sup>, HARALD VRANKEN<sup>1,2</sup>, ANDRÉ ZÚQUETE<sup>3</sup>, AND ALI MIRI<sup>4</sup>

<sup>1</sup>Department of Computer Science, Open Universiteit, 6401 Heerlen, The Netherlands

<sup>2</sup>Institute for Computing and Information Sciences, Radboud University, 6500 Nijmegen, The Netherlands

<sup>3</sup>Instituto de Engenharia Electrónica e Informática de Aveiro (IEETA), University of Aveiro, 3810-193 Aveiro, Portugal

<sup>4</sup>Department of Computer Science, Ryerson University, Toronto, ON M5B 2K3, Canada

Corresponding author: Hassan Alizadeh (hassan.alizadeh@ou.nl)

This work was supported in part by the Portuguese National Funds through FCT—Foundation for Science and Technology, in the context of Ph.D. Scholarship under Grant SFRH/BD/84037/2012, and in part by the Open University of the Netherlands. The APC was funded by the Open University of the Netherlands.

**ABSTRACT** We present a novel approach for timely classification and verification of network traffic using Gaussian Mixture Models (GMMs). We generate a separate GMM for each class of applications using component-wise expectation-maximization (CEM) to match the network traffic distribution generated by these applications. We apply our models for both traffic classification, where the goal is to identify the source application from which the traffic originates, by evaluating the maximum posterior probability, and for traffic verification, where the goal is to verify whether the application that claims to be the source of the traffic is as expected, by likelihood testing. Our models use only the first initial packets of truncated flows in order to provide more efficient and timely traffic classification and verification. This allows for triggering timely countermeasures before the end of flows. We demonstrate the effectiveness of our approach by experiments on a public dataset collected from a real network. Our traffic classification approach outperforms other state-of-the-art approaches that are based on machine learning, and achieves up to 97.7% flow classification accuracy when using only 9 first initial packets of flows. We show that 96.6% flow classification accuracy can still be obtained when training the GMMs using only 0.5% of all flows. Our traffic verification approach achieves a minimum Half Total Error Rate (HTER) of 7.65% when using only 6 first initial packets of flows.


**INDEX TERMS** Gaussian mixture model (GMM), traffic classification, traffic anomaly detection.

## I. INTRODUCTION

The number of internet applications and the variety of end-users is increasing continuously, as well as the number of online network attacks and advanced generations of malware. This has caused that both *classification* and *verification* of network traffic have become more difficult. The task of traffic classification is to identify the source application from which the traffic originates, while the task of traffic verification is to verify whether the application that claims to be the source of the traffic is as expected. Accurate classification and verification is essential, particularly for network operators in network management tasks such as resource allocation, accounting, traffic scheduling, quality of service, lawful interception of IP traffic, and network security. Continuous research on both

traffic classification and traffic verification is required to mitigate these problems.

In this paper, we present a novel generative approach for timely traffic classification and traffic verification. The key idea in our approach is that we generate a separate Gaussian Mixture Model (GMM) for each application class, where an application class is a group of related applications such as e-mail clients. Before generating the GMM for an application class, we first derive feature vectors from the network traffic generated by the applications in the class. The feature vectors include features such as the number of packets in network flows, and statistics on the size of packets and the interarrival times between packets. Subsequently, we apply the feature vectors in a training stage to generate the GMM for the application class. The GMM is a probabilistic model that represents the probability density function of the feature vectors. Hence, the GMM models the probabilistic

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier García Villalba .

distribution of feature vectors for that application class. We apply component-wise expectation-maximization (CEM) to generate the GMM [1].

Once the GMMs are generated, we can apply them for both traffic classification and traffic verification. For traffic classification, we monitor network flows in the network traffic, we extract feature vectors, and evaluate which of the GMMs correlates best with these feature vectors by means of a maximum a posteriori probability (MAP) estimate. For traffic verification, we consider network traffic that is labelled with its application class, and we verify whether it matches the GMM of the specified application class by means of likelihood testing.

Our approach can tackle the problem of concept drift in network traffic, which occurs for instance when the number of applications is growing in time. We can simply generate GMMs for the new application classes and add them to the set of GMMs, without the need for keeping all past training data and retraining GMMs. Hence, our approach facilitates smooth adaption for changing traffic profiles.

This paper contributes the following:

- 1) We extend our prior work on traffic classification and verification and present in more detail the parameters estimation and model selection when training the GMMs [2].
- 2) We present a novel set of 59 statistical features that are derived from the first initial packets of truncated flows. We obtain optimal feature subsets for different numbers of first initial packets of flows for both traffic classification and traffic verification, and explore their effectiveness for generating GMMs.
- 3) We present experimental results to demonstrate that we are able to classify and verify network traffic in a timely way. This may for instance facilitate early detection of zero-day attacks.
- 4) We demonstrate that our approach outperforms state-of-the-art approaches for traffic classification that are based on machine learning.

The rest of the paper is organized as follows. In Section II we review prior work on traffic classification and traffic verification. In section III we present our approach to construct GMMs that subsequently can be applied for GMM-based traffic classification and verification. In Section IV we present the details of our experimental setups and evaluation results. We conclude the paper in Section V.

## II. LITERATURE REVIEW

### A. TRAFFIC CLASSIFICATION

Traffic flows play a key role in traffic classification. Network traffic generated by different individual applications running on one or multiple hosts can be aggregated and separated by traffic flows. Each flow can be defined as a (unidirectional or bidirectional) sequence of IP packets sharing typically a 5-tuple identifier (source IP address, destination IP address, source port, destination port, protocol type) within a certain

period of time. The aim of traffic classification is to map each specific flow to a group of applications of interest.

Traditional classification approaches are port-based or payload-based. In the port-based approach, applications are simply identified by the TCP or UDP port numbers. However, many applications, such as P2P applications and Skype, are increasingly using arbitrary port numbers to avoid detection, which downgrades classification accuracy and renders the use of port-based identification suboptimal. In the payload-based approach, applications are identified by the actual payloads of packets or the reassembled contents of flows, looking for characteristic signatures of known applications. Payload-based approaches provide high accuracy, however they require high computational cost, privacy is a major concern, and they are likely to fail with encrypted payloads [3].

An alternative approach is to rely on statistical characteristics of traffic flows, regardless of their payloads, which can overcome the limitations of port-based and payload-based approaches [4]. The main assumption behind this approach is that the statistical characteristics (features or discriminators) of network traffic flows are distinct for different applications and can be used to distinguish applications from each other. A lightweight approach that is not hampered by encrypted payloads, is to consider statistical characteristics extracted from packet headers only while ignoring the packet payloads. Relying on completed flows, which requires to wait for the end of flows, is less practical or suboptimal in cases where quick and timely classification is required. This can be relaxed by considering statistical characteristics from truncated flows, in which only the first few packets of flows are considered. It has been shown by multiple researchers that such truncated flows may provide sufficiently distinctive properties that allow to distinguish applications from each other in an early stage of traffic flows [5]–[11]. However, most prior work only considered simple features such as size, direction, and sequence of the first non-zero payload packets of flows [5], [6], [8], [9]. Furthermore, they did not consider TCP control features, such as SYN and ACK, and statistical features, such as mean and variance of packet size and packet interarrival time. Consequently, attackers can evade such classification by simply padding the packet payloads. Li and Moore [7] extracted more sophisticated features from the first few packets of flows, starting from the SYN-packet up to a duration of 5 seconds. However, their features also included the port number pairs, and hence classification results may be biased towards port numbers. Consequently, attackers can evade such classification by performing port masquerading. Liu *et al.* [10] used 11 statistical features from the first few packets of flows. Garcia and Korhonen [11] captured distributional characteristics of the initial packets of flows considering histogram fractions.

The use of machine learning for traffic classification has been explored extensively, partly due to the availability of off-the-shelf learning algorithms and their efficiency in dealing with statistical information [3]. Several challenges and requirements have to be considered [12]: an appropriate

definition of a set of classes is required to specify the classes of network traffic that should be distinguished from each other (e.g. P2P versus web traffic [13]) or to identify specific application protocols [14]; tools are required to extract (near real-time) flow features that have the potential to be deployed in online real-world settings (e.g. TIE [15] and Netmate-flowcalc); benchmarking tools (e.g. [16], [17]) or network architectures (e.g. [18]–[21]) are required that generate 'reliable ground truth' [16], [22] aiming at evaluating various traffic classification approaches [23]; measurements should provide byte accuracy in addition to flow accuracy [24]; and the temporal stability of classification schemes should be examined [25]. Most prior work has focused on discriminative algorithms, including Support Vector Machines (SVM) [26] and Decision Trees (DT) [7], where a single conditional distribution model is trained for all classes and used next to discriminate the classes from each other [27]. This however may suffer from concept drift when the number of applications is constantly growing in time. The model has to be updated with the advent of a new class, which requires retraining for which also all past training data likely need to be accessible. This makes discriminative approaches in their current forms suboptimal.

More recently also deep learning has been applied for traffic classification and anomaly detection [28]–[31]. Different deep learning methods vary in their performance, but most of them provide high accuracy and are also well able to detect previously unknown attacks. However, more training data is required to gain sufficient accuracy, and also the time and computational power required to train models is in general much larger when compared to machine learning. If the primary goal is to reduce training time and computational costs, as we consider in this paper, deep learning methods are not preferred.

Also the use of GMM as a generative approach for traffic classification has been explored in prior work [6], [32], [33]. Bernaille *et al.* [6] proposed a method in which a single GMM is used to approximate the underlying feature distributions of all classes. Moore and Zuev [32] proposed to model the underlying feature distributions of each class by a Gaussian distribution, using a single component for all the classes. Dusi *et al.* [33] proposed an internet traffic classifier that utilizes class-specific GMMs to approximate the underlying feature distribution of each (type of) application injecting traffic on SSH-encrypted tunnels. These methods assign any unseen observation to the class that maximizes the posterior probability.

Regardless of whether traffic classification is based on class-conditional GMMs or a single class-independent GMM, a model selection technique is needed to provide good performance. Prior GMM-based traffic classifiers build a set of candidate GMMs with different numbers of components and then select the one that provides the best performance [6], [33]. However, training an entire set of GMMs along with the EM-associated problems is a main drawback (see Section III-B2).

The approach proposed by Figueiredo and Jain [34] automatically selects the optimal number and shape of mixture components in a single algorithm. The semi-supervised traffic classification framework presented by Qian *et al.* [35] utilizes the GMM learning approach proposed by Figueiredo and Jain [34]. However, they only consider a single GMM that models all network traffic, and each GMM component is assigned to an application. Our approach also adopts the GMM learning approach proposed by Figueiredo and Jain [34], but to the best of our knowledge, our approach is the first attempt to automatically find the best GMM for each individual application in traffic classification.

## B. TRAFFIC VERIFICATION

Network traffic anomalies are inferred from patterns in network traffic data that deviate from (or do not conform to) the normal network behaviour (profiles) [36]. Signature-based intrusion detection systems match observed traffic against a pre-configured set of (known) intrusion signatures, while anomaly-based intrusion detection systems notice deviations from pre-defined models (profiles) describing normal traffic. The latter may detect new types of (still unknown) intrusions and zero-day attacks, but have to be adapted constantly due to changing profiles. García-Teodoro *et al.* presented an overview of various anomaly-based intrusion detection systems [37].

Anomaly-based detection relies on the assumption that the statistical characteristics of traffic differ between normal and anomalous traffic. Statistical characteristics of normal network traffic are used to build users' profiles. These statistical characteristics can be extracted from one or more traffic features, such as packet header, payload, flow, bandwidth usage or packet distribution.

Payload-based approaches [38] are probably one of the most promising ways of detecting traffic anomalies that originate from infected applications. These methods have the ability to provide near real-time detection with high accuracy levels. However, as for payload-based classification, also payload-based verification suffers from high computational costs, privacy concerns, and difficulties when dealing with encrypted or otherwise securely encapsulated traffic [39]. Alternative approaches utilise the information available in the non-encrypted IP packet header and analyse statistical characteristics of flows regardless of packet payload.

Much of the research on anomaly-based detection has been devoted to build network normality profiles from aggregated network traffic, and flows are a fundamental object of study [40], [41]. Prior flow-based and payload-based approaches may be able to detect some intrusion-infected traffic, but they are unable to identify the source application responsible for the traffic. Moreover, they cannot detect abnormal traffic generated by an intrusion-infected application whenever such traffic is very similar to the normal traffic of other known applications. To address this problem a detection system needs to evaluate whether traffic is normal for its source application. An anomaly-based detection

system therefore should meet two requirements: (i) identify the claimant (source application) that is responsible for the traffic, and (ii) model or profile the genuine traffic of each application present in the network. To meet the first requirement, we proposed architectures that provide a binding between network traffic and source application that allows checking whether a packet/flow claimed by an application conforms to its expected traffic model [42], [43]. For the second requirement, we proposed GMM with automatic learning to model per-application traffic [2], [39], [44]. However, our prior application-specific models were still trained with features obtained from the entire packet flows and were not accurate enough for detecting anomalies in a timely manner before the end of a flow. Moreover, this approach suffered from the difficulty of finding an optimal number of mixture components. We tackled the latter issue by applying GMM learning [2], but the problem of timely detection remained to be addressed. Also Bahrololum and Khaleghi [45] applied GMM in anomaly-based detection, however they only considered full flows. Since sophisticated intrusions may meet their illicit purposes through one single flow, timely detection and prevention is important. In the present paper, we address this issue and explore the effectiveness of truncated flows for different numbers of initial packets in order to provide more efficient and timely detection.

### III. GMM DESIGN

We apply a GMM-based approach for both traffic classification and verification. In this section we first describe the GMM used and its parameterization. Next, we describe traffic classification and verification.

#### A. MODELLING APPLICATIONS

GMMs are an effective way for data analysis and system modelling, including those involving random processes. A GMM is a probabilistic model for representing the probability density function of a population of data points as a weighted mixture of probability density functions of normally distributed sub-populations (components). For each application class  $a$  we compute a separate GMM using traffic packets from truncated flows generated by the application(s) in the class. We extract  $D$ -dimensional feature vectors  $x$  from these flows, and model their probability distribution by a GMM with  $M$  components. The GMM of an application class  $a$  is defined as:

$$p(x|\Theta^a) = \sum_{m=1}^M \omega_m^a p(x|\theta_m^a) \quad (1)$$

where  $\Theta^a = \{\theta_1^a \dots \theta_M^a, \omega_1^a \dots \omega_M^a\}$  is the set of all parameters in the model. Here,  $\omega_m^a$  is a mixture weight that satisfies the constraints  $\omega_m^a > 0$  and  $\sum_{m=1}^M \omega_m^a = 1$ , and  $\theta_m^a$  is a set of parameters containing the mean  $D \times 1$  vector  $\mu_m^a$  and the  $D \times D$  covariance matrix  $\Sigma_m^a$ . The unimodal Gaussian density

function of component  $m$  is defined as:

$$p(x|\theta_m^a) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_m^a|}} e^{-\frac{1}{2}(x-\mu_m^a)^T (\Sigma_m^a)^{-1} (x-\mu_m^a)} \quad (2)$$

Here,  $|\Sigma_m^a|$  denotes the determinant of the covariance matrix,  $(\Sigma_m^a)^{-1}$  denotes the inverse covariance matrix, and  $T$  denotes transposition. The covariance matrix can either be free (unrestricted) or diagonal (restricted). Additionally, the covariance matrix can be constrained to be the same across mixture components.

#### B. PARAMETERS ESTIMATIONS

For a given set of  $N$  feature vectors  $X = \{x_1, x_2, \dots, x_N\}$  that is extracted from (truncated) flow samples generated by an application class  $a$ , the GMM is defined as (assuming that the feature vectors are independent, which may be incorrect but assumed to make the problem tractable):

$$p(X|\Theta^a) = \prod_{n=1}^N p(x_n|\Theta^a) \quad (3)$$

The set of feature vectors is used as a training set to construct the GMM. The goal is to estimate the parameters in  $\Theta^a$  such that the GMM matches the distribution of the training samples. These parameters estimates can be obtained using either the *Maximum Likelihood (ML)* estimate

$$\hat{\Theta}_{ML}^a = \arg \max_{\Theta^a} \{\log p(X|\Theta^a)\} \quad (4)$$

or the *Maximum A Posteriori (MAP)* estimate (given some prior  $p(\Theta^a)$ )

$$\hat{\Theta}_{MAP}^a = \arg \max_{\Theta^a} \{\log p(X|\Theta^a) + \log p(\Theta^a)\} \quad (5)$$

through an *Expectation-Maximization (EM)* algorithm.

##### 1) THE EM ALGORITHM

With feature vector  $x_n$  we associate a label  $z_n$ , which is an  $M$ -dimensional vector indicating which of the  $M$  GMM components produces  $x_n$ . The EM algorithm interprets  $X$  as incomplete data, with the missing part being the corresponding set of labels  $Z = \{z_1, \dots, z_N\}$  in which  $z_n = [z_{n,1}, \dots, z_{n,M}]$  is a binary vector with  $z_{n,m} = 1$  and  $\forall_{p \in \{1, \dots, M\} \setminus m} : z_{n,p} = 0$ , meaning that  $x_n$  was produced by the  $m^{th}$  GMM component. Assuming that all  $x_n$  are independent, the complete data log-likelihood function for the GMM can be written as [46], [47]:

$$\log p(X, Z|\Theta^a) = \sum_{n=1}^N \sum_{m=1}^M z_{n,m} \log [\omega_m^a p(x_n|\theta_m^a)] \quad (6)$$

The EM algorithm begins with an initial estimate of parameters  $\hat{\Theta}^a(0)$  and iteratively generates a sequence of estimates  $\{\hat{\Theta}^a(t)|t = 1, 2, \dots\}$  by alternatively applying two steps, namely Expectation (or E-Step) and Maximization (or M-Step), as follows:

E-Step computes the expected value of the conditional probability of the log-likelihood, given the observed data  $X$

and the current parameter estimates  $\hat{\Theta}^a(t)$ , yielding the following  $Q$ -function:

$$Q(\Theta^a, \hat{\Theta}^a(t)) \equiv \mathbb{E} \left[ \log p(X, Z | \Theta^a) | X, \hat{\Theta}^a(t) \right] = \log p(X, \Psi | \Theta^a) \quad (7)$$

where  $\Psi \equiv \mathbb{E} \left[ Z | X, \hat{\Theta}^a(t) \right]$ . The latter equality is obtained from the linearity of  $\log p(X, Z | \Theta^a)$  with respect to  $Z$  (see Equation 6). Since the members of  $Z$  are binary, their conditional expectations can be expressed as the posterior probability that  $x_n$  was produced by the  $m^{\text{th}}$  GMM component for application class  $a$ :

$$\psi_{n,m}^a \equiv \mathbb{E} \left[ z_{n,m} | X, \hat{\Theta}^a(t) \right] = \Pr \left[ z_{n,m} = 1 | x_n, \hat{\Theta}^a(t) \right] = \frac{\hat{\omega}_m^a(t) p(x_n | \hat{\theta}_m^a(t))}{\sum_{j=1}^M \hat{\omega}_j^a(t) p(x_n | \hat{\theta}_j^a(t))} \quad (8)$$

M-Step updates the parameter estimates using a MAP estimation according to:

$$\hat{\Theta}^a(t+1) = \arg \max_{\Theta^a} \left\{ Q(\Theta^a, \hat{\Theta}^a(t)) + \log p(\Theta^a) \right\} \quad (9)$$

The term  $\log p(\Theta^a)$  is removed in the case of ML estimation.

The EM algorithm is iterated until either the changes in the estimated parameters, the number of iterations, or the log-likelihood exceed some specified threshold (whichever comes first).

## 2) ESTIMATING THE NUMBER OF COMPONENTS

Since the data distribution likely differs for different application classes, exploring the best fit for each individual application seems worthwhile. One of the key choices in training a GMM is choosing the number of components  $M$ . The general trend observed in literature is to apply a deterministic approach that first builds a set of candidate models for a range of values of  $M$  (from  $M_{\min}$  to  $M_{\max}$ ), and next selects from the entire set of available models the one that best fulfils a given selection criterion such as the Bayesian Inference Criterion (BIC) [48], the Minimum Description Length (MDL) [49], or the Minimum Message Length (MML) [50]. Although some of these approaches may perform well, building a whole set of candidate models is a major drawback. Moreover, since they essentially utilize the EM algorithm for building the models, they are prone to two major EM-associated problems: sensitivity to initialization of the parameters, meaning that different initial values of parameters may converge to different local optima, and not necessarily the global one, during the maximization process (M-Step); and possible convergence of the sequence of EM parameter estimates to the boundary of the parameter space (where the likelihood is boundless).

To deal with these problems, we adopt the approach proposed by Figueiredo and Jain [34], which facilitates a seamless integration of parameters estimation and model selection. We assume that the number of components of each application class is unknown and can be different for different

application classes. The method implements a variant of the EM algorithm, named component-wise EM (CEM) [1], with the aim of minimizing the following MML criterion as the cost function:

$$\Gamma(\Theta^a, X) = \frac{K}{2} \sum_{m=1}^M \log \left( \frac{N \omega_m^a}{12} \right) + \frac{M_{n,z}}{2} \log \left( \frac{N}{12} \right) + \frac{M_{n,z}(K+1)}{2} - \log p(X | \Theta^a) \quad (10)$$

where  $K = D + D(D+1)/2$  is the number of parameters specifying each component and  $M_{n,z} \in \{1, 2, \dots, M\}$  denotes the number of components whose probability is nonzero. The cost function, for fixed  $M_{n,z}$ , is mathematically equivalent to an a posteriori probability density function yielding from a Dirichlet-type prior for  $\omega_m^a$  and a flat prior for  $\theta_m^a$ 's. With some adjustments to minimize the cost function, the sequence of parameter estimates of the M-step is summarized as follows:

$$\hat{\omega}_m^a(t+1) = \frac{\max \left\{ 0, \left( \sum_{n=1}^N \psi_{n,m}^a \right) - \frac{K}{2} \right\}}{\sum_{j=1}^M \max \left\{ 0, \left( \sum_{n=1}^N \psi_{n,j}^a \right) - \frac{K}{2} \right\}} \quad (11)$$

$$\hat{\mu}_m^a(t+1) = \frac{\sum_{n=1}^N x_n \psi_{nm}^a}{\sum_{n=1}^N \psi_{n,m}^a} \quad (12)$$

$$\hat{\Sigma}_m^a(t+1) = \frac{\sum_{n=1}^N x_n^2 \psi_{nm}^a}{\sum_{n=1}^N \psi_{n,m}^a} - (\hat{\mu}_m^a(t+1))^2 \quad (13)$$

for  $m = 1, 2, \dots, M$  and  $\psi_{n,m}$  given in the E-step. With the use of Equation 11 and allowing  $M$  to move from a large number  $M_{\max}$  to a smaller one  $M_{\min}$ , the algorithm not only tackles the problem of 'convergence to the boundary of parameter space' by annihilating 'too weak' components, but also becomes less sensitive to the initialization problem [51]. However, the simultaneous (or batch) updating of  $\hat{\omega}_m^a$ 's may cause a failure in the determination of  $\omega_m^a$ 's. For a large value of  $M$ , all components may not have enough initial support ( $\sum_{n=1}^N \psi_{n,m}^a < K/2$  for  $m = 1, \dots, M$ ) and, consequently, all  $\omega_m^a$ 's cannot be determined. This problem is avoided by the use of CEM. CEM performs the updating of all  $\hat{\omega}_m^a$  and  $\hat{\theta}_m^a = \{\hat{\mu}_m^a, \hat{\Sigma}_m^a\}$  sequentially, rather than simultaneously. In other words, CEM executes E-step for any single updating of  $\hat{\omega}_m^a$  and  $\hat{\theta}_m^a$ , rather than for batch updating of all parameter estimates. This allows immediate redistribution of the probability mass of a zero-component to the other components. The CEM iteration is continued until the relative decrease in  $\Gamma(\Theta^a(t), X)$  drops below a given threshold  $\varepsilon$ . This convergence results in a certain value for  $M$ . However, a smaller value of  $M_{n,z}$  may lead to additional decrease in the value of  $\Gamma(\Theta^a, X)$ . For this reason, after reaching convergence with

a certain  $M$ , the component with smaller  $\hat{\omega}_m^a$  is set to zero and CEM is rerun until convergence. This process is repeated while  $M_{n,z} \geq M_{\min}$ . At the end, the estimated parameters as well as the number of components are those which minimize  $\Gamma(\Theta^a, X)$ .

### C. TRAFFIC CLASSIFICATION

In traffic classification we apply the GMMs and assign each observed traffic flow to the most probable application class. Given a set of  $K$  application classes  $Y = a_1, a_2, \dots, a_K$  represented by the models  $\Theta^{a_1}, \Theta^{a_2}, \dots, \Theta^{a_K}$ , our objective is to determine the application model which has the maximum posterior probability for feature vector  $x$ . Hence, we assign each flow to the class with the highest  $p(\Theta^{a_i}|x)$ :

$$\hat{y} = \arg \max_{i \in \{1, \dots, K\}} p(\Theta^{a_i}|x) \quad (14)$$

Using Bayes' rule, Equation 14 evolves into:

$$\hat{y} = \arg \max_{i \in \{1, \dots, K\}} \frac{p(x|\Theta^{a_i})p(\Theta^{a_i})}{p(x)} \quad (15)$$

The term  $p(x|\Theta^{a_i})$  is computed as in Equation 1, and  $p(\Theta^{a_i})$  refers to the prior probability of application  $a_i$  that is derived in the training stage when generating the GMM for application  $a_i$ . The term  $p(x)$  is constant for all applications and can be ignored.

### D. TRAFFIC VERIFICATION

In traffic verification we decide whether traffic is normal or abnormal for its source application. In the latter case, we consider the traffic to contain an anomaly. Given a (truncated) flow sample  $x$  along with its supposed source application  $a_i$ , we estimate whether or not  $x$  was generated by  $a_i$  by considering model  $\Theta^{a_i}$  for application  $a_i$ . We perform a likelihood test by evaluating  $p(x|\Theta^{a_i}) \geq \tau$ , which reflects whether traffic sample  $x$  matches the normal traffic of its supposed source application  $a_i$  given threshold  $\tau$ .

The verification decision can be erroneous in two ways: *False Acceptance (FA)* and *False Rejection (FR)*. FA occurs when an actual anomaly in the traffic is considered as normal traffic; FR occurs when traffic from a clean application is considered as an anomaly. The *False Acceptance Rate (FAR)* is the percentage of anomalies that are considered as normal (i.e.,  $p(x_{\text{abnormal}}|\Theta^{a_i}) \geq \tau$ ); the *False Rejection Rate (FRR)* is the percentage of clean samples that are considered as abnormal (i.e.,  $p(x_{\text{normal}}|\Theta^{a_i}) < \tau$ ). We determine the optimal value for  $\tau$  based on the *Equal Error rate (EER)* criterion where FAR is equal to FRR, using an evaluation set from the traffic data. We consider two scenarios for determining  $\tau$ : the *Global Threshold (GT)* employs a single threshold for all classes; the *Class-Specific Threshold (CST)* employs separate thresholds for each individual class.

## IV. EXPERIMENTAL METHODOLOGY

We performed various experiments to explore the effectiveness of our GMM-based approach for traffic classification

and traffic verification. In this section we first describe in detail the dataset used in our experiments. Next, we describe the experimental setup and evaluation metrics. Finally, we present the experimental results.

### A. DATASETS DESCRIPTION

We conducted our experiments on the *UNIBS-2009* dataset.<sup>1</sup> This dataset was collected from 20 workstations in the campus network of the University of Brescia in Italy on three consecutive working days (September 30, October 1, and October 2, 2009). The data was collected by running tcpdump on the edge router that connects the network to the Internet via a dedicated 100Mb/s uplink. The dataset consists of four files: three PCAP files (containing 27 GB data) and a logfile (groundtruth.log).

#### 1) FLOW DEFINITION

The traffic contains around 79,000 flows that were generated by various types of applications employing web protocols (HTTP and HTTPS), mail protocols (POP3, IMAP4, SMTP, and their SSL variants), Skype, peer-to-peer, and other protocols (FTP, SSH, and MSN).

We focused on the TCP traffic that represents more than 98% of the data in the UNIBS dataset. A flow is defined by a set of bidirectional consecutive packets traveling between two endpoints (defined by source IP address, source port number, destination IP address, and destination port number) through a TCP connection started by a 3-way handshake (SYN, SYN-ACK, and ACK) and terminated by either observing FIN/RST packets or 'no packet seen' for a timeout of 60 s (whichever comes first). The first SYN-packet seen in a flow determines the forward direction, and the source endpoint is assigned as the client. We consider only TCP flows that have at least one packet in each direction and contain at least one non-zero payload packet.

#### 2) FEATURE EXTRACTION

A flow associated with a particular application can be described by a number of statistical properties, or features, parameterizing its behaviour. We modified and employed the Netmate-flowcalc tool<sup>2</sup> to extract the bidirectional flows as defined above, and calculated their statistical feature values from the PCAP files. We checked the order of packets within each flow and ignored out-of-order packets during the feature calculation process. In order to also support truncated flows, we modified Netmate-flowcalc to extract the statistical features of a flow in a PCAP trace after observing the  $n^{\text{th}}$  packet counting from the SYN-packet. We refer to  $n$  as the *Reference Packet Count (RPC)*.

In prior studies on traffic classification using truncated flows, Garcia and Korhonen [11] obtained best accuracy for RPC values ranging from 12 to 15. However, they applied distributional flow features for the classification of video traffic,

<sup>1</sup><http://www.ing.unibs.it/ntw/tools/traces/>

<sup>2</sup><http://sourceforge.net/projects/netmate-meter/files/netmate-meter/>

which only partially relates to our context. Peng *et al.* [8] experimented with RPC values ranging from 2 to 10, but they only considered packets with non-zero payload. They concluded that using 5 to 7 non-zero payload packets is most effective, since including more packets reduces the time performance of a classifier, while including less packets reduces the accuracy. They also concluded that the effective number of packets varies with network environments. Liu *et al.* [10] experimented with RPC values ranging from 5 to 10 for the classification of network traffic using statistical features, which more closely resembles our context. They observed that classification accuracy gradually increased when ranging the number of packets from 5 to 8, and slightly decreased when taking 9 and 10 packets. Based on these prior studies, we range RPC values from 3 to 10 in our experiments. We do not consider RPC values larger than 10, also since we aim at timely classification and verification of truncated flows. For comparison, we also explore results obtained when considering complete flows. For each specific value of *RPC*, we built a separate dataset. Some short-lived flows may not reach the *RPC*. We still considered such flows in our analysis, since they may carry infected payloads.

In total we derived 59 flow features. Table 1 lists the full feature set, including their abbreviations and indexes.<sup>3</sup> Most of these features are calculated separately for the forward and backward direction. In our experiments the features are extracted from the packets within the RPC range. Features that correspond to packets outside the RPC range (such as features 40-59 that specify the length of individual packets send in the forward direction or the backward direction) are ignored. The statistical features are recomputed when a new packet in a flow is encountered. The features include no information about the payload, and hence the feature set is content-independent, which removes privacy concerns, avoids dealing with encrypted payloads, and provides that features can be calculated at low computational cost. The features also include no information about IP addresses and port numbers, and hence the feature set is site-independent and robust against port-masquerading strategies in which an (unknown) application evades port-based detection by simply changing its destination port number to that of a well-known application.

### 3) GROUND TRUTH

The logfile in the UNIBS dataset contains all TCP and UDP flows and their starting time associated with the source applications and protocols. Application-level ground truth was assigned to the traffic flows through the architecture proposed by Gringoli *et al.* [21]. Flows are assigned to the following application categories: *Web Browsers*, *MAILS*, *P2P*, *SKYPE*, and *OTHERS*.

<sup>3</sup>An idle time of 1 second was used to distinguish sub-flows. A sub-flow ends when no packet is received within 1 second; the next new sub-flow starts with the next packet.

TABLE 1. Features.

Index	Description	Abbreviation
1	total forward packets	total-fpackets
2	total forward volume	total-fvolume
3	total backward packets	total-bpackets
4	total backward volume	total-bvolume
5	min forward packet length	min-fpktl
6	mean forward packet length	mean-fpktl
7	max forward packet length	max-fpktl
8	standard deviation of forward packet length	std-fpktl
9	min backward packet length	min-bpktl
10	mean backward packet length	mean-bpktl
11	max backward packet length	max-bpktl
12	standard deviation of backward packet length	std-bpktl
13	min forward interarrival time	min-fiat
14	mean forward interarrival time	mean-fiat
15	max forward interarrival time	max-fiat
16	standard deviation of forward interarrival time	std-fiat
17	min backward interarrival time	min-biat
18	mean backward interarrival time	mean-biat
19	max backward interarrival time	max-biat
20	standard deviation of backward interarrival time	std-biat
21	duration of flow	duration
22	min active time	min-active
23	mean active time	mean-active
24	max active time	max-active
25	standard deviation of active times	std-active
26	min idle time	min-idle
27	mean idle time	mean-idle
28	max idle time	max-idle
29	standard deviation of idle time	std-idle
30	sub-flow forward packets	sflow-fpackets
31	sub-flow forward bytes	sflow-fbytes
32	sub-flow backward packets	sflow-bpackets
33	sub-flow backward bytes	sflow-bbytes
34	forward push counter	fpsh-cnt
35	backward push counter	bpsh-cnt
36	forward urgent counter	furg-cnt
37	backward urgent counter	burg-cnt
38	total forward header length	total-fhlen
39	total backward header length	total-bhlen
40	length of 1st packet in forward direction	fpl1
41	length of 2nd packet in forward direction	fpl2
42	length of 3rd packet in forward direction	fpl3
43	length of 4th packet in forward direction	fpl4
44	length of 5th packet in forward direction	fpl5
45	length of 6th packet in forward direction	fpl6
46	length of 7th packet in forward direction	fpl7
47	length of 8th packet in forward direction	fpl8
48	length of 9th packet in forward direction	fpl9
49	length of 10th packet in forward direction	fpl10
50	length of 1st packet in backward direction	bpl1
51	length of 2nd packet in backward direction	bpl2
52	length of 3rd packet in backward direction	bpl3
53	length of 4th packet in backward direction	bpl4
54	length of 5th packet in backward direction	bpl5
55	length of 6th packet in backward direction	bpl6
56	length of 7th packet in backward direction	bpl7
57	length of 8th packet in backward direction	bpl8
58	length of 9th packet in backward direction	bpl9
59	length of 10th packet in backward direction	bpl10

Table 2 summarizes the TCP flows/packets/bytes in the UNIBS dataset per day and per application category. We explicitly show the break-down per day since we use temporal information in our experiments. For instance, we built our GMMs with data from the first day and evaluated the GMMs with data from the next two days. This kind of evaluation clearly separates data for model training and evaluation, and is more practical for real-world scenarios.

**TABLE 2.** The UNIBS-2009 dataset: TCP flows/packets/bytes break-down by application category and day.

Application Category	Day 1			Day 2			Day 3			All Days			Examples of protocols
	Flows	Packets	Bytes	Flows	Packets	Bytes	Flows	Packets	Bytes	Flows	Packets	Bytes	
MAILS (Apple Mail, Thunderbird)	1,473	58.9K	11M	1,813	73.9K	16.7M	1,615	62K	12M	4,901	195K	40M	IMAP, POP3, SMTP, SSL
P2P (Transmission, eMule, Bittorrent)	5,121	2.6M	1.59G	1,114	1.86M	1.3G	15,113	23.4M	20G	21,348	27.9M	22.9G	edonkey, bittorrent, HTTP
WEB Browsers (Safari, Firefox, Opera)	9,811	919K	837M	18,424	1.14M	932M	16,880	1.68M	1.4G	45,115	3.7M	3.2G	HTTP, HTTPS
SKYPE	379	33.6K	3.3M	334	128K	100M	341	193K	150M	1,054	355K	254M	skype, HTTP
OTHERS	586	31.7K	20.7M	1,237	87.8K	74.5M	884	39.5K	28M	2,707	159K	123M	FTP, SSH, MSN
Total	17,370	3.7M	2.46G	22,922	3.3M	2.45G	34,833	25.4M	21.68G	75,125	32.4M	26.6G	

#### 4) FEATURE SELECTION

We applied feature selection to find a subset of the full feature set that yields the highest performance at the lowest computational cost. Finding the optimal feature subset is a NP-hard problem and diverse feature selection approaches have been proposed in the machine learning literature [52], [53]. We adopted the Sequential Forward Selection (SFS) algorithm for selecting the optimal feature subset. The SFS algorithm takes a greedy approach. It starts with an empty feature set and gradually adds features to the feature set. At each iteration, the feature is selected from the candidate feature set that optimizes an evaluation function. This feature is then added to the feature set and removed from the candidate set. The algorithm continues until the addition of further features does not improve the evaluation function. We selected the SFS algorithm since we aim at minimizing computational cost. The complexity of the SFS algorithm is  $O(n^2)$ , where  $n$  is the number of features. More advanced feature selection algorithms could have provided better results, but at larger time complexity [54], [55].

### B. EXPERIMENTAL SETUP AND EVALUATION METRICS

#### 1) MODEL DESIGN SETUP

We trained the GMMs using the MATLAB-code as provided by Figueiredo and Jain [34].<sup>4</sup> In both traffic classification and verification experiments, we allowed the covariance matrices to be free (unrestricted) when training the GMMs. We experimentally observed that unrestricted covariance matrices outperformed covariance matrices that are either diagonal (restricted) or constrained to be the same across mixture components. We set the regularizing factor for the covariance matrices and the stopping threshold  $\varepsilon$  for CEM iteration to  $10^{-4}$ . For both parameters, we experimentally found that performance was rather insensitive to values in the range from  $10^{-6}$  to  $10^{-2}$  and degraded for values larger than  $10^{-2}$ . We set the initial minimum and maximum number of mixture components ( $M_{min}$  and  $M_{max}$ ) to 1 and 10, respectively.

#### 2) TRAFFIC CLASSIFICATION EXPERIMENTS

In our traffic classification experiments we used the flows from Day 1 for training and the flows from Days 2 and 3 for testing. We trained 4 models for the application classes

*MAILS*, *P2P*, *WEB Browsers*, and *SKYPE* (see Table 2). We excluded the applications class *OTHERS* during training since it does not contain traffic of a coherent set of related applications. We also excluded this class during testing, and hence our classification experiments resemble a best-case scenario in which only traffic is considered originating from applications that the models have been trained for.

In our experiments we compared the performance of our GMM-based approach with 6 of the most commonly used supervised machine learning algorithms [56]: Naive-Bayes (NB) [32], [57], [58], Decision Tree (DT) [7],  $k$ -Nearest Neighbour ( $k$ -NN) [59], Support Vector Machines (SVM) [26], [60], [61], Random Forest (RF) [11], [62], [63], and Gradient Boosted Trees (GBT) [63]. We used scikit-learn<sup>5</sup> to implement all classifiers, except for GBT we used XGBOOST<sup>6</sup> as it is significantly faster. For the NB and DT classifiers we used the default values. For the  $k$ -NN classifier, we employed  $k = 1$  since it achieved the highest overall accuracy among  $k = 3, 5, 7, 9$ . For the SVM classifier, we employed LibSVM [64] with *RBF* Kernel function. Yuan *et al.* [60] showed that this kernel function achieved the best traffic classification accuracy among three other kernel functions, namely *LINEAR*, *POLY*, and *SIGMOID*. For the RF and GBT classifiers, we applied RandomizedSearchCV in scikit-learn to perform a randomized search over 6 hyper-parameters, including a range of 10 to 500 for the number of trees. The search involved 2-fold cross-validation over a set of 20 different hyper-parameter settings. We used the average F1-measure to select the best setting, and applied this setting for training the classifier.

Our experiments included the following three steps:

- 1) We applied the SFS algorithm as shown in Algorithm 1 in order to select the best feature subset. In each iteration we apply  $k$ -fold cross-validation in which the training dataset is randomly split into  $k$  (approximately) equal disjoint subsets. Each of the subset acts in turn as the evaluation set for the models trained with the other subsets. We apply 2-fold cross-validation to reduce computational cost. At the end of each iteration we add the feature that maximizes the average F1-measures of the models.

<sup>5</sup><https://scikit-learn.org/>

<sup>6</sup>[https://xgboost.readthedocs.io/en/latest/python/python\\_api.html](https://xgboost.readthedocs.io/en/latest/python/python_api.html)

<sup>4</sup>[www.lx.it.pt/~mtf/mixturecode2.zip](http://www.lx.it.pt/~mtf/mixturecode2.zip)



**Algorithm 1** SFS for Feature Selection in Traffic Classification

---

```

1:  $selected \leftarrow \emptyset$  // set of selected features
2:  $candidates \leftarrow$  set of 59 features shown in Table 1
3:  $y \leftarrow$  labels from dataset (ground truth)
4:  $F_{post} \leftarrow 0$  // average F1-measure
5:  $k \leftarrow 2$  // k-fold cross validation
6: repeat
7:    $F_{pre} \leftarrow F_{post}$ 
8:   for all  $f \in candidates$  do
9:      $features \leftarrow selected \cup \{f\}$ 
10:    split  $dataset$  into  $k$  subsets  $data_1 \dots data_k$ 
11:     $\hat{y} \leftarrow \emptyset$  // observed classification
12:    for  $i = 1$  to  $k$  do
13:      train models with  $dataset \setminus data_i$  and  $features$ 
14:       $\hat{y}_i \leftarrow$  test models with  $data_i$ 
15:       $\hat{y} \leftarrow \hat{y} \cup \hat{y}_i$ 
16:    end for
17:     $F_{post}^f \leftarrow evaluate(\hat{y}, y)$ 
18:  end for
19:   $f \leftarrow$  feature yielding  $max_{f \in candidates}(F_{post}^f)$ 
20:   $F_{post} \leftarrow F_{post}^f$ 
21:  if ( $F_{post} > F_{pre}$ ) then
22:     $selected \leftarrow selected \cup \{f\}$ 
23:     $candidates \leftarrow candidates \setminus \{f\}$ 
24:  end if
25: until ( $F_{post} \leq F_{pre}$ )
26: train models with  $selected$  features and  $dataset$ 
27: test models with  $test\ dataset$ 

```

---

We applied the SFS algorithm for every method (GMM, NB, DT,  $k$ -NN, SVM RF, GBT) and all  $RPC$  values ranging from 3 to 10. All further classification experiments were conducted using the selected feature subsets.

- 2) We evaluated the performance of the models on the level of correctly classified *flows*, *packets* and *bytes*, using the *Overall Accuracy* and the *average F1-measure* over all classes as measures. The *Overall Accuracy* determines the percentage of all *flows/packets/bytes* that are correctly classified. For per-class measures, *Precision* is the percentage of *flows/packets/bytes* correctly classified to a class over the total number of *flows/packets/bytes* assigned to that class, and *Recall* is the percentage of *flows/packets/bytes* from a given class that are truly classified to that class. We consider the *F1-measure* which takes the harmonic mean of both precision and recall as a single accuracy measure (i.e.,  $2 \times (precision \times recall) / (precision + recall)$ ).
- 3) We examined the impact of the training set size on the classification performance of our GMM-based approach.

## 3) TRAFFIC VERIFICATION EXPERIMENTS

In our traffic verification experiments using our GMM-based approach we considered three main stages, namely *training*, *evaluation*, and *testing*. We used the flows from Day 1 for training, the flows from Day 2 for evaluation, and the flows from Day 3 for testing. We excluded the applications class *OTHERS* during training, since it does not contain traffic of a coherent set of related applications. We considered all application classes, including the *OTHERS* class, during evaluation and testing. Hence our verification experiments resemble a scenario in which also traffic is considered originating from applications that the models have not been trained for.

We applied the algorithm as outlined in Algorithm 2. We applied the SFS algorithm to find the optimal subset of features and determined the optimal value for the threshold  $\tau$ , while aiming to minimize the Equal Error Rate, using the training dataset and the evaluation dataset. As indicated in section III-D, we determined both the *Global Threshold (GT)* over all classes, and *Class-Specific Thresholds (CST)* for each individual class. In the testing stage, we tried different values of  $RPC$  in order to find the optimal  $RPC$  value. We report the *Half Total Error Rate (HTER)*, which takes the average of  $FAR$  and  $FRR$  as a single measurement.  $FAR$ ,  $FRR$ ,  $EER$  and  $HTER$  range from 0% (best) to 100% (worst).

**Algorithm 2** SFS for Feature Selection and Threshold Setting in Traffic Verification

---

```

1:  $selected \leftarrow \emptyset$  // set of selected features
2:  $candidates \leftarrow$  set of 59 features shown in Table 1
3:  $E_{post} \leftarrow 100\%$  // EER
4: repeat
5:    $E_{pre} \leftarrow E_{post}$ 
6:   for all  $f \in candidates$  do
7:      $features \leftarrow selected \cup \{f\}$ 
8:     train GMMs with  $training\ dataset$ 
9:      $(\tau^f, E_{post}^f) \leftarrow$  evaluate GMMs with
        $evaluation\ dataset$ 
10:   end for
11:    $f \leftarrow$  feature yielding  $min_{f \in candidates}(E_{post}^f)$ 
12:    $E_{post} \leftarrow E_{post}^f$ 
13:   if ( $E_{post} < E_{pre}$ ) then
14:      $selected \leftarrow selected \cup \{f\}$ 
15:      $candidates \leftarrow candidates \setminus \{f\}$ 
16:      $\tau \leftarrow \tau^f$ 
17:   end if
18: until ( $E_{post} \geq E_{pre}$ )
19: test GMMs with  $test\ dataset$ 

```

---

**C. EXPERIMENTAL RESULTS**1) TRAFFIC CLASSIFICATION RESULTS  
OPTIMAL FEATURE SUBSETS

Figure 1 shows the optimal feature subsets obtained by the SFS algorithm for the different methods and  $RPC$  values. It can be seen that the number of selected features per subset

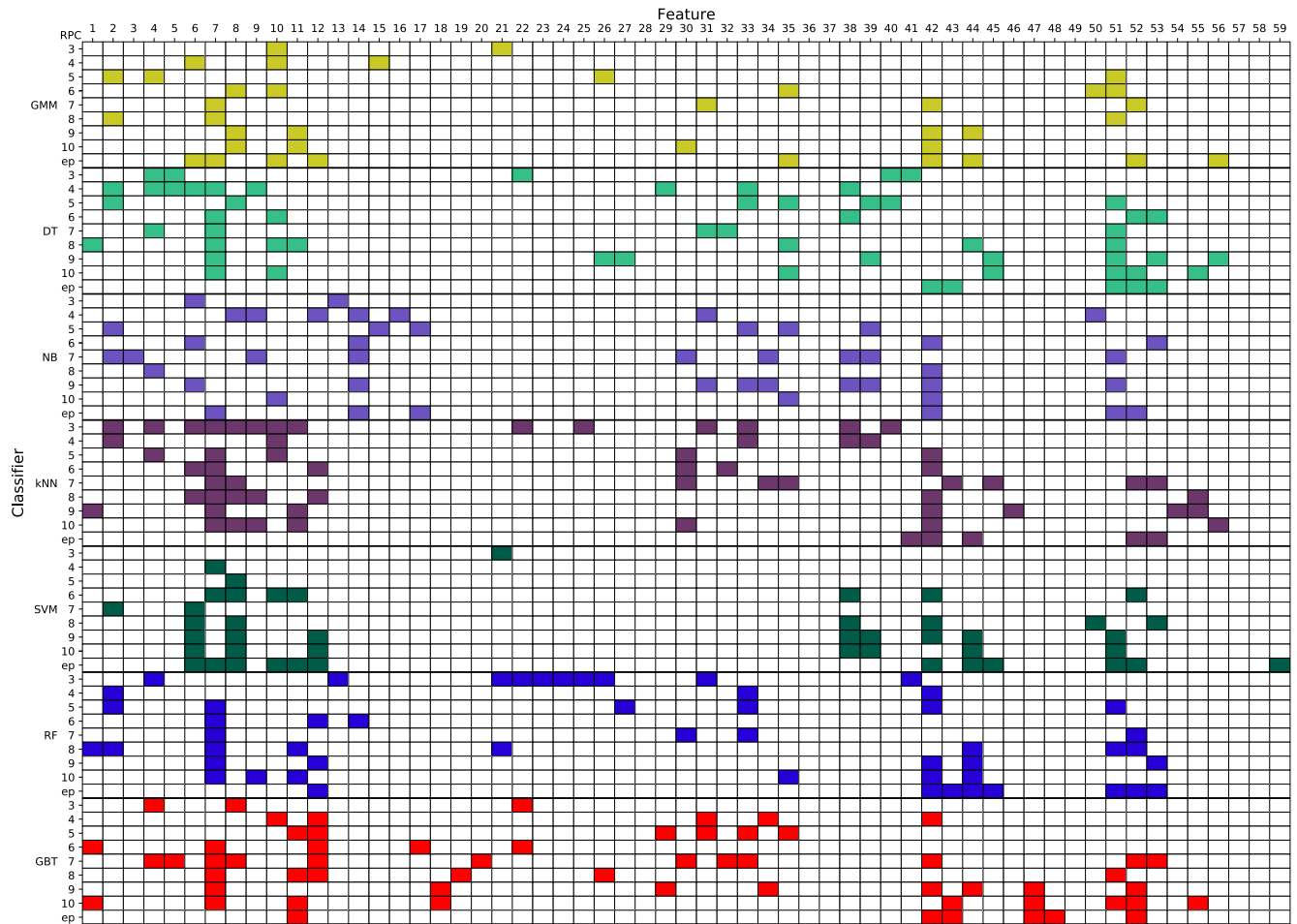


FIGURE 1. Best feature subsets obtained for each dataset (RPC) using the SFS algorithm. (See Table 1 for the features.)

varies lightly, but the selected features per subset vary considerably, not only per method but also per RPC value. The features 7 and 42 are selected most often and are included in half of the subsets, while ten features are included only in a single subset and six features are not selected at all.

The optimal feature subsets provide sufficient classification capabilities. As an example, we illustrate the distinctive capability of the optimal feature subset for the GMM models for  $RPC = 9$  in an 'Andrews plot' [65] and a 'parallel coordinates plot' [66] shown in Figure 2. The Andrews plot represents each feature vector  $x$  as a Fourier series where the coefficients are equal to the feature values. The parallel coordinates plot represents each feature vector by the sequence of features plotted against their values. The figure shows the feature vectors from 250 randomly selected flows for each class from Day 1 of the UNIBS dataset. The Fourier series in the example Andrews plot has 5 terms over the interval  $[0,1]$ : a constant, two sine terms with periods  $1/2$  and  $1$ , and two similar cosine terms. The variations in these plots demonstrate that the feature subset sufficiently discriminates the different application categories.

Hereafter, we perform the classification experiments using the optimal feature subsets as shown in Figure 1.

a: IMPACT OF RPC ON PERFORMANCE

Figure 3 compares the performance results of different methods, including the proposed GMM approach, in terms of Overall Accuracy and average F1-measure at the level of flows, packets, and bytes.

Figure 3a illustrates the Overall Accuracy at the level of flows for different methods as a function of the  $RPC$ . It is clear that by increasing  $RPC$ , the performance of all approaches except NB improves, which indicates that using more data packets for extracting the flow features provides higher Overall Accuracy. The improvement of GMM is significant and GMM yields the highest Overall Accuracy when  $RPC$  ranges from 4 to 10, clearly outperforming all other methods. The Overall Accuracy with GMM increases up to 97.74% when  $RPC$  increases from 3 to 9, and slightly decreases for larger  $RPC$ . When using complete flows, RF achieves the highest Overall Accuracy.

Figure 3b compares the average F1-measures at the flow level for the different methods. Also here, best results are obtained with GMM when  $RPC$  ranges from 4 to 10, except for  $RPC$  is 6. NB and SVM do not perform well. The performance of all approaches lacks behind when  $RPC$  ranges from 3 to 6. We observed that this is mostly due to poor

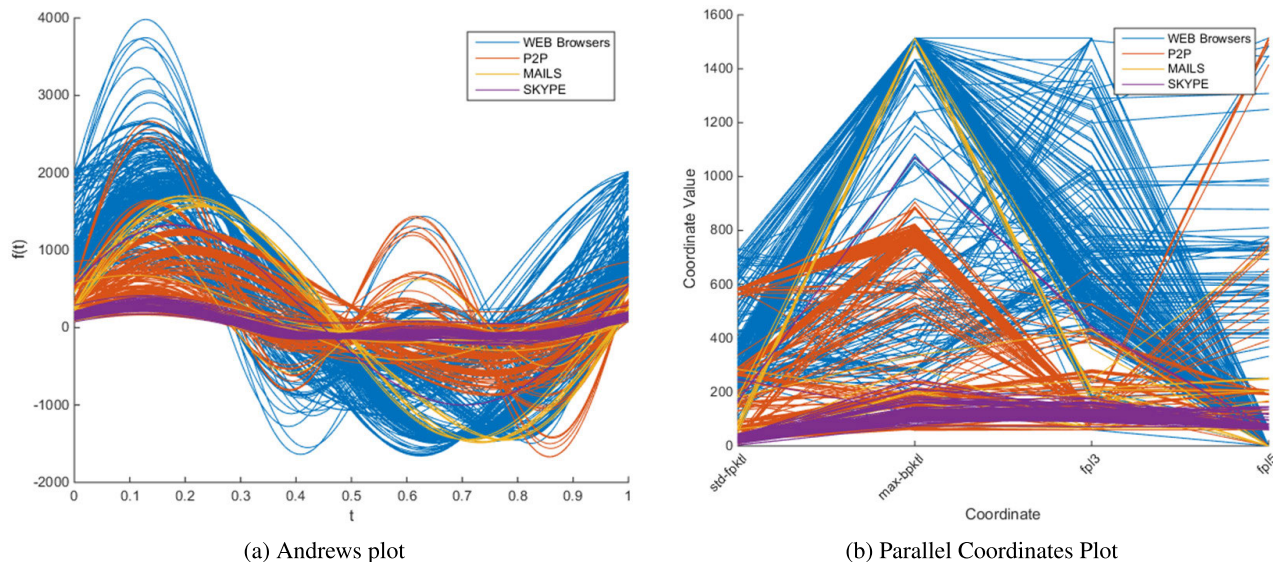


FIGURE 2. Andrews plot (left) and Parallel coordinates plot (right) for feature set of GMMs with  $RPC = 9$ .

performance for at least one rare class, mostly *SKYPE*. This can be seen in Table 3a, where the classification results for all classes are shown when  $RPC = 9$ . The average F1-measures with GMM increases up to 93.66% when  $RPC$  increases from 3 to 10, and slightly increases when using complete flows. RF achieves the highest average F1-measures for complete flows.

Figure 3c illustrates the Overall Accuracy at the level of bytes for different methods as a function of  $RPC$ . We see that GMM still performs well, but not as outstanding as for the level of flows as shown in Figure 3a. This can be explained by the observation that GMM improved the Overall Accuracy at the flow level by providing better classification of 'mice flows' (mostly belonging to *Web Browsers*) than other classifiers. Flows with small and large payload sizes are referred as 'mice' and 'elephant' flows, respectively [24].

Figure 3d compares the average F1-measures at the byte level for different methods. GMM provides the best results when  $RPC$  is 9 (68.74%) and 10 (69.30%). The F1-measures for all approaches is (far) below 70%. These rather low F1-measure values can be explained by the fact that all approaches have poor performance at the byte level for at least one class. This can be seen in Table 3b, which compares performance at the byte level for all methods when  $RPC = 9$ . It is clear that the poor performance results for *SKYPE* hampered the overall performance results of all classifiers. We observed that this is due to miss-classification of five elephant flows responsible for 96% of all bytes in the test subset.

Figure 3e and 3f illustrate the Overall Accuracy and the average F1-measure at the level of packets for different methods as a function of  $RPC$ . We observe a behaviour similar to the one at the byte level.

In summary, we conclude that the proposed GMM approach outperforms all other methods at the flow level

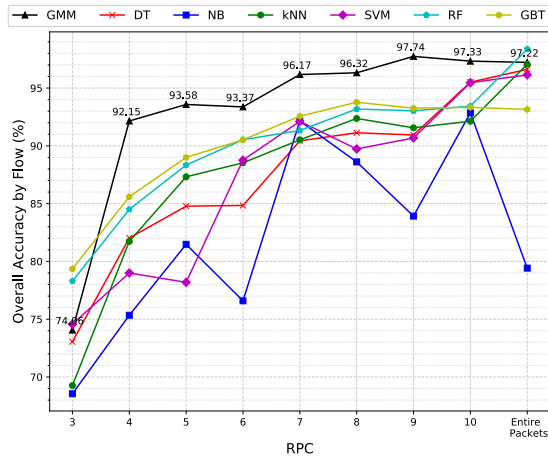
when  $RPC$  ranges from 7 to 10, both for the Overall Accuracy and average F1-measures. The maximum Overall Accuracy (97.74%) is achieved for  $RPC = 9$ , and the maximum average F1-measures (93.66%) is achieved for  $RPC = 10$ . At the level of bytes and packets, best results are obtained with GMM when  $RPC$  ranges from 9 to 10, while RF and GBT perform almost equally well. The maximum Overall Accuracy (98.28% and 98.38%) as well as the maximum average F1-measures (69.30% and 79.57%) is achieved for  $RPC = 10$ .

Table 3 shows the performance of the different classifiers for  $RPC = 9$  per class in terms of Overall Accuracy and F1-measure for flows, bytes, and packets, respectively.

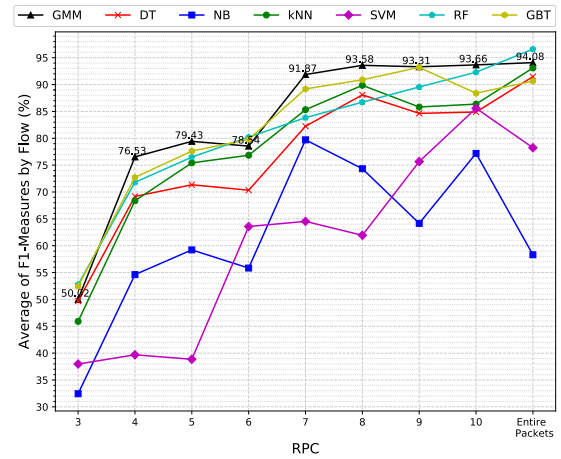
*b: IMPACT OF TRAINING SET SIZE*

Since preparing sufficiently large-scale, well-labelled training data is laborious, it is useful to determine how much training data is needed for a desired accuracy. For this reason, we evaluated the sensitivity of the proposed GMM approach for the size of the training set when  $RPC = 9$ .

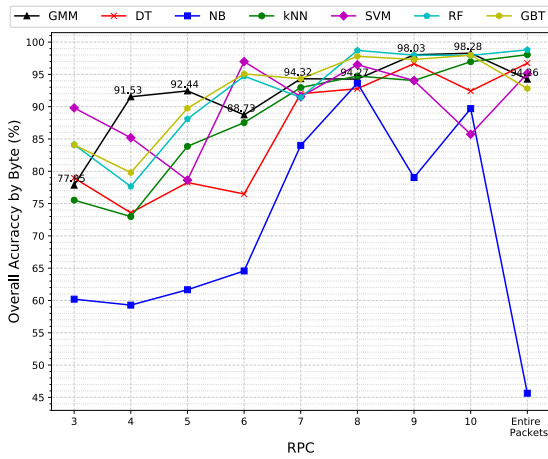
Figure 4 presents the Overall Accuracy as the number of the training flows varies (on a logarithmic scale). Each mark is the average result of running the GMM models 10 times. For each run, the models are trained on a different training set of the same size that is randomly selected from Day 1, and tested on the total number of flows obtained from the combination of Day 2 and Day 3. The error-bars in the figure show the standard deviation of the mean. The figure shows that the performance steadily improves and stabilizes as the training set size increases. We observed that 96.58% and 96.89% Overall Accuracy can be achieved with 403 and 1,097 randomly selected flows as training set, respectively, which constitute around 0.5% and 1.5% of all flows. Such small training sets greatly reduce



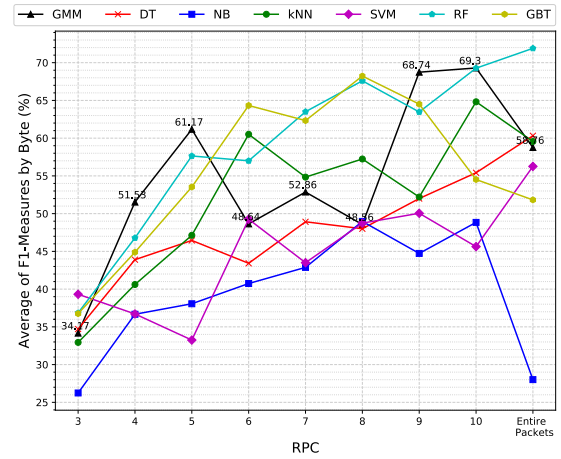
(a) Overall accuracy by flows



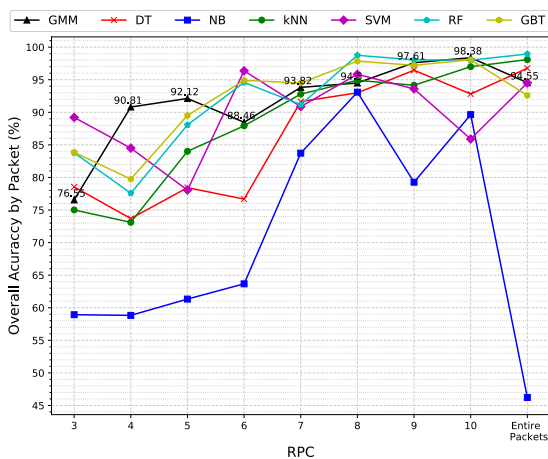
(b) Average F1-measure by flows



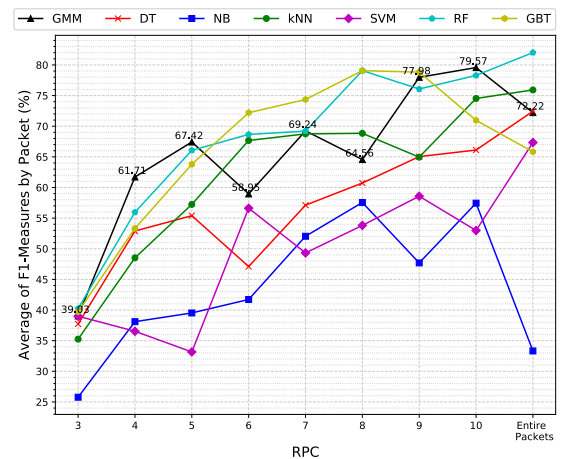
(c) Overall accuracy by bytes



(d) Average F1-measure by bytes



(e) Overall accuracy by packets



(f) Average F1-measure by packets

**FIGURE 3.** Assessment of the classifiers through their Overall Accuracy and Average F1-measure by Flow, Byte, and Packet for different RPC values.

the time required to build the models. We also observed that the classification speed was approximately constant at  $3.7 \times 10^5$  flows/second. These observations make the

proposed approach promising for practical uses, where well-labelled training data is scarce or re-training the models is a serious concern.

**TABLE 3. Performance of the different classifiers for  $RPC = 9$  per class in terms of Overall Accuracy and F1-measure by flow, byte, and packet (best results are marked in bold).**

Class	GMM	DT	NB	k-NN	SVM	RF	GBT
Mails	92.21	89.35	52.20	92.50	63.90	94.14	<b>96.12</b>
P2P	<b>97.74</b>	87.25	82.99	87.87	94.35	90.05	89.99
Web Browsers	<b>98.51</b>	93.65	92.71	93.97	93.48	94.81	94.72
SKYPE	84.78	68.30	28.76	68.96	50.91	79.21	<b>91.97</b>
Average F1 (Flows)	<b>93.31</b>	84.64	64.16	85.82	75.66	89.55	93.20
Overall Accuracy	<b>97.74</b>	90.94	83.92	91.56	90.68	93.03	93.26

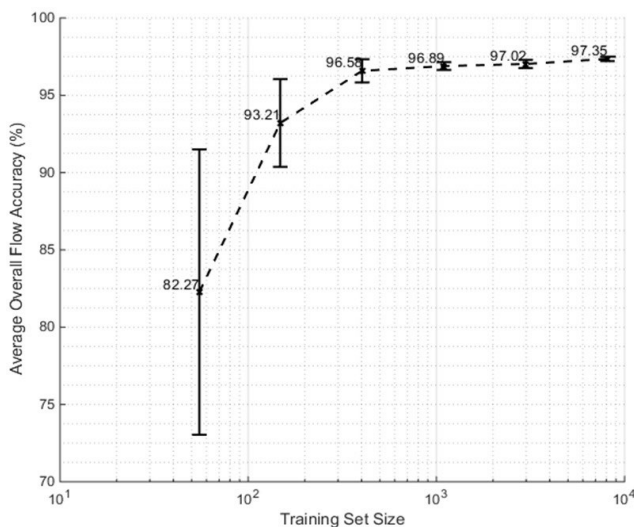
(a) Flows

Class	GMM	DT	NB	k-NN	SVM	RF	GBT
Mails	<b>78.50</b>	12.34	2.53	24.83	13.64	56.09	64.42
P2P	<b>99.48</b>	98.71	88.09	97.17	97.72	99.45	99.07
Web Browsers	92.48	91.99	87.85	84.27	88.08	<b>94.61</b>	87.97
SKYPE	4.49	4.95	0.45	2.52	0.77	3.73	<b>6.57</b>
Average F1 (Bytes)	<b>68.74</b>	52.00	44.73	52.20	50.05	63.47	64.51
Overall Accuracy	<b>98.03</b>	96.66	79.03	94.05	94.04	97.98	97.31

(b) Bytes

Class	GMM	DT	NB	k-NN	SVM	RF	GBT
Mails	<b>91.54</b>	39.06	12.33	60.23	44.06	84.37	88.83
P2P	99.15	98.50	88.05	97.13	97.30	<b>99.40</b>	98.90
Web Browsers	90.71	91.14	86.68	84.32	86.94	<b>94.81</b>	87.49
SKYPE	30.54	31.48	3.70	18.11	6.00	25.71	<b>40.04</b>
Average F1 (Packets)	77.98	65.04	47.69	64.95	58.58	76.07	<b>78.82</b>
Overall Accuracy	97.61	96.46	79.27	94.15	93.59	<b>98.05</b>	97.21

(c) Packets



**FIGURE 4. Average overall accuracy vs. size of training set.**

2) TRAFFIC VERIFICATION RESULTS

Table 4 lists the *CST*-dependent and *GT*-dependent feature subsets selected by the SFS algorithm for different values of *RPC*. These results were obtained using the evaluation subset for different values of *RPC* aiming at minimizing *EER*.

Figures 5a and 5b compare performance results of our GMM-based traffic verification approach for different values of *RPC* with both the *CST* and the *GT* thresholds.

**TABLE 4. *CST*-dependent and *GT*-dependent feature subsets selected by the SFS algorithm.**

RPC	<i>CST</i> -dependent Selected features	<i>GT</i> -dependent Selected features
3	4, 6, 9, 21, 38	10, 22, 28, 33, 40
4	2, 42	4, 6
5	5, 7, 40, 52	7, 35, 41, 52
6	5, 42	7, 9, 11, 40
7	39, 40, 42	38, 41, 42
8	3, 40, 42	42, 50
9	30, 41, 42	38, 42, 50
10	40, 42, 56	41, 42
All Packets	5, 42	10, 42, 50

**TABLE 5. Class-specific *HTER* of *CST* and *GT* on the test subset when *RPC* is 8.**

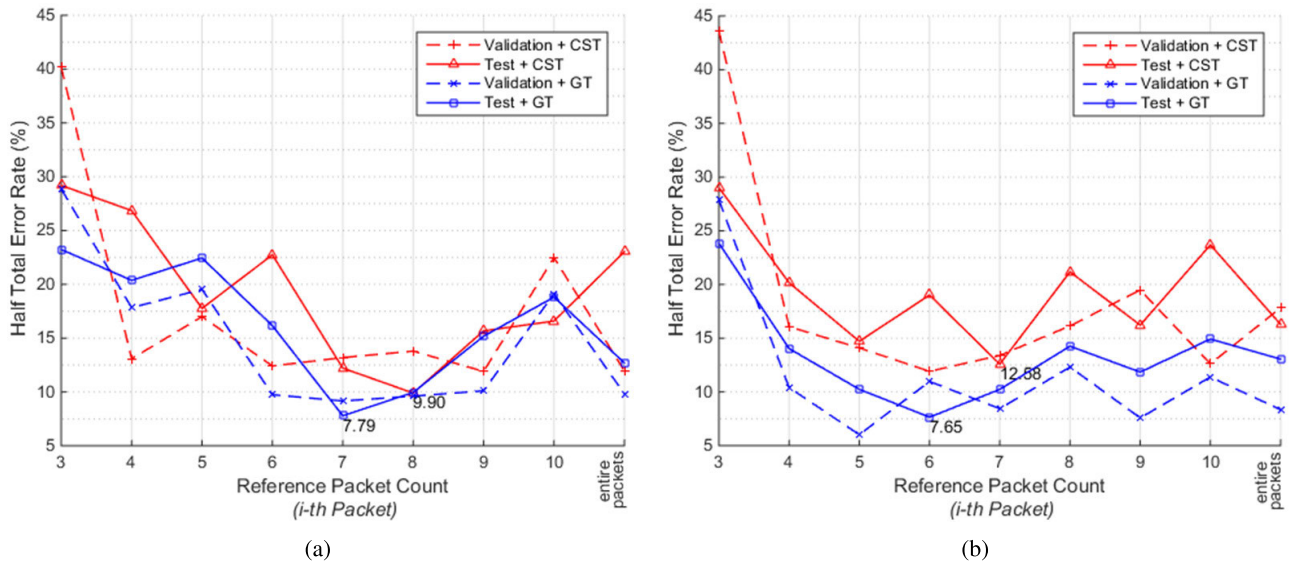
Class	<i>HTER</i> (%) by <i>CST</i>	<i>HTER</i> (%) by <i>GT</i>
MAILS	4.74	4.67
P2P	4.56	4.49
Web Browsers	12.84	23.37
SKYPE	11.48	14.36

Figure 5a shows results when using the *CST*-dependent feature subsets (given in second column of Table 4); Figure 5b shows results when using the *GT*-dependent feature subsets (given in third column of Table 4). Both figures compare *HTER* for both the evaluation and the test subset.

The figures show that using *GT* provides better results than using *CST*. This is particularly the case when using the *GT*-dependent feature subsets as shown in Figure 5b, where *GT* outperforms *CST* for all *RPC* values with both the evaluation and the test subset. A *HTER* of less than 15% is achieved with *GT* for all values of *RPC*, except for *RPC* = 3. Even when using the *CST*-dependent feature subsets, shown in Figure 5a, *GT* still outperforms *CST* in most cases, except for *RPC* of 4 and 5 on the evaluation subset, and for *RPC* of 5 and 10 on the test subset.

On the test set, a minimum *HTER* of 7.79% is achieved with *GT* for *RPC* = 7 using the *CST*-dependent feature subsets (Figure 5a), and 7.65% for *RPC* = 6 using the *GT*-dependent feature subsets (Figure 5b).

Table 5 compares *HTER* for the test set per class when using the *CST*-dependent feature set and *RPC* = 8, where a minimum *HTER* of 9.9% is achieved by both *GT* and *CST* (see Figure 5a). It clearly shows that while both *CST* and *GT* achieved a comparable *HTER* for *MAILS* and *P2P*, *CST* significantly outperformed *GT* for *Web Browsers* and *SKYPE*. This is as expected, since the goal of *CST* is to minimize *HTER* of each individual application, while the goal of *GT* is to minimize the overall *HTER* of all applications combined.



**FIGURE 5.** HTER by CST and GT as a function of RPC for both the evaluation and the test subset, using the CST-dependent (left) and GT-dependent (right) selected features as shown in Table 4.

Hence, while *CST* yields a better *HTER* per application, *GT* still yields a better overall *HTER*. This is due to the fact that the number of likelihood scores available to determine the optimal *GT* threshold is larger than the number of likelihood scores available for determining the optimal *CST* thresholds.

## V. CONCLUSION

In this paper we presented an almost real-time traffic classification as well as an equally fast application-aware traffic anomaly detection system based on an original use of GMMs. The learning algorithm used, unlike the basic Expectation-Maximization (EM) algorithm, selects an optimal number of mixture components automatically with a seamless integration of estimating mixture parameters from given multivariate data. The representation of each application (type) is provided by a GMM fitted to the underlying distribution of flow-level features of that application. The traffic classification approach assigns any flow to the class with the highest posterior probability. The traffic verification approach for anomaly detection is based on class-specific and global thresholding mechanisms, where a threshold is set at the *EER* operating point to determine whether a flow claimed by an application is genuine. In order to provide a timely operation, only the first initial packets of flows are considered in the learning process. We adopted the SFS feature selection algorithm for selecting the optimal feature subset. We evaluated the effectiveness of our GMM-based approaches by conducting different sets of experiments on a public dataset collected from a real network. In order to provide efficient and timely traffic classification and anomaly detection, the effectiveness of different numbers of first initial packets of flows has been explored and evaluated. Our traffic classification approach considerably improves on other state-of-the-art approaches that are based

on machine learning. Our GMM-based approach achieves an Overall Accuracy for flows of 97.74% using 9 initial packets of flows. We observed that 96.6% and 96.9% Overall Accuracy at the flow level, respectively, can be maintained by only using 0.5% and 1.5% of all flows for training the GMMs. Our GMM-based anomaly detection achieved a minimum Half Total Error Rate (HTER) of 7.65% by using only 6 initial packets of flows.

In our future work we intend to extend our GMM-based approach such that it also can be applied in an online non-stationary environment, where both class evolution and concept drift occur in time. We also consider to evaluate our approach with other datasets (when available), and to evaluate related approaches, such as Variational Bayesian model selection [67], as well as deep learning methods.

## REFERENCES

- [1] G. Celeux, S. Chretien, F. Forbes, and A. Mkhadri, "A component-wise EM algorithm for mixtures," *J. Comput. Graph. Statist.*, vol. 10, no. 4, pp. 697–712, Dec. 2001.
- [2] H. Alizadeh, A. Khoshrou, and A. Zuquete, "Traffic classification and verification using unsupervised learning of Gaussian mixture models," in *Proc. IEEE Int. Workshop Meas. Netw. (MN)*, Oct. 2015, pp. 1–6.
- [3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 2008.
- [4] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *Int. J. Netw. Manage.*, vol. 25, no. 5, pp. 355–374, Sep. 2015.
- [5] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, Apr. 2006.
- [6] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proc. ACM CoNEXT Conf. (CoNEXT)*. New York, NY, USA: ACM, 2006, pp. 6:1–6:12.
- [7] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," in *Proc. 15th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Oct. 2007, pp. 310–317.

- [8] L. Peng, B. Yang, and Y. Chen, "Effective packet number for early stage Internet traffic identification," *Neurocomputing*, vol. 156, pp. 252–267, May 2015.
- [9] J.-H. Ham, H.-M. An, and M.-S. Kim, "Application traffic classification using PSS signature," *KSH Trans. Internet Inf. Syst.*, vol. 8, no. 7, pp. 2261–2280, 2014.
- [10] Y. Liu, J. Chen, P. Chang, and X. Yun, "A novel algorithm for encrypted traffic classification based on sliding window of flow's first N packets," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Appl. (ICCIA)*, Sep. 2017, pp. 463–470.
- [11] J. Garcia and T. Korhonen, "Efficient distribution-derived features for high-speed encrypted flow classification," in *Proc. Workshop Netw. Meets AI ML (NetAI)*. New York, NY, USA: ACM, 2018, pp. 21–27.
- [12] A. Dainotti, A. Pescapé, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan. 2012.
- [13] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between Web and peer-to-peer traffic in the network core," in *Proc. 16th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2007, pp. 883–892.
- [14] A. Este, F. Gringoli, and L. Salgarelli, "On the stability of the information carried by traffic flow features at the packet level," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 3, pp. 13–18, Jun. 2009.
- [15] W. De Donato, A. Pescapé, and A. Dainotti, "Traffic identification engine: An open platform for traffic classification," *IEEE Netw.*, vol. 28, no. 2, pp. 56–64, Mar. 2014.
- [16] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Comput. Netw.*, vol. 76, pp. 75–89, Jan. 2015.
- [17] M. Canini, W. Li, A. W. Moore, and R. Bolla, "GTVS: Boosting the collection of application traffic ground truth," in *Traffic Monitoring and Analysis*. Berlin, Germany: Springer, 2009, pp. 54–63.
- [18] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," in *Passive and Active Network Measurement (Lecture Notes in Computer Science)*, vol. 4979. Berlin, Germany: Springer, 2008, sec. 8, pp. 72–81.
- [19] P. Lizhi, Z. Hongli, Y. Bo, C. Yuehui, and W. Tong, "Traffic labeller: Collecting Internet traffic samples with accurate application information," *China Commun.*, vol. 11, no. 1, pp. 69–78, Jan. 2014.
- [20] B. Lee, S. Moon, and Y. Lee, "Application-specific packet capturing using kernel probes," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Jun. 2009, pp. 303–306.
- [21] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: Picking up the truth from the ground for Internet traffic," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 5, pp. 12–18, Oct. 2009.
- [22] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, "Is our ground-truth for traffic classification reliable?" in *Passive and Active Measurement (Lecture Notes in Computer Science)*, vol. 8362. Cham, Switzerland: Springer, 2014, sec. 10, pp. 98–108.
- [23] S. Lee, H. Kim, D. Barman, S. Lee, C.-K. Kim, T. Kwon, and Y. Choi, "NeTraMark: A network traffic classification benchmark," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 22–30, Jan. 2011.
- [24] J. Erman, A. Mahanti, and M. Arlitt, "Byte me: A case for byte accuracy in traffic classification," in *Proc. 3rd Annu. ACM Workshop Mining Netw. Data (MineNet)*. New York, NY, USA: ACM, 2007, pp. 35–38.
- [25] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, no. 6, pp. 790–809, Apr. 2009.
- [26] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Comput. Netw.*, vol. 53, no. 14, pp. 2476–2490, Sep. 2009.
- [27] C. M. Bishop and J. Lasserre, "Generative or discriminative? Getting the best of both worlds," *Bayesian Statist.*, vol. 8, pp. 3–24, Jan. 2007.
- [28] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, no. S1, pp. 949–961, Jan. 2019.
- [29] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2020.
- [30] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.
- [31] T. Bodström and T. Hämläinen, "State of the art literature review on network anomaly detection with deep learning," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems (Lecture Notes in Computer Science)*, vol. 11118. Cham, Switzerland: Springer, 2018, pp. 64–76.
- [32] A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 50–60, Jun. 2005.
- [33] M. Dusi, A. Este, F. Gringoli, and L. Salgarelli, "Using GMM and SVM-based techniques for the classification of SSH-encrypted traffic," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–6.
- [34] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, Mar. 2002.
- [35] F. Qian, G.-M. Hu, and X.-M. Yao, "Semi-supervised Internet network traffic classification using a Gaussian mixture model," *AEU-Int. J. Electron. Commun.*, vol. 62, no. 7, pp. 557–564, Aug. 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1434841107001409>
- [36] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
- [37] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, Feb. 2009.
- [38] A. Jamdagni, Z. Tan, X. He, P. Nanda, and R. P. Liu, "RePIDS: A multi tier real-time payload-based intrusion detection system," *Comput. Netw.*, vol. 57, no. 3, pp. 811–824, Feb. 2013.
- [39] H. Alizadeh, S. Khoshrou, and A. Zúquete, "Application-specific traffic anomaly detection using universal background model," in *Proc. ACM Int. Workshop Int. Workshop Secur. Privacy Anal. (IWSPA)*. New York, NY, USA: ACM, 2015, pp. 11–17, doi: [10.1145/2713579.2713586](https://doi.org/10.1145/2713579.2713586).
- [40] R. Hofstede, V. Bartos, A. Sperotto, and A. Pras, "Towards real-time intrusion detection for NetFlow and IPFIX," in *Proc. 9th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2013, pp. 227–234.
- [41] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *Proc. 4th IFIP Int. Conf. New Technol., Mobility Secur.*, Feb. 2011, pp. 1–5.
- [42] A. Zuquete, P. Correia, and H. Shamalazadeh, "Packet tagging system for enhanced traffic profiling," in *Proc. IEEE 5th Int. Conf. Internet Multimedia Syst. Archit. Appl.*, Dec. 2011, pp. 1–6.
- [43] A. Zuquete and M. Rocha, "Identification of source applications for enhanced traffic analysis and anomaly detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 6694–6698.
- [44] H. Alizadeh and A. Zúquete, "Traffic classification for managing applications' networking profiles," *Secur. Commun. Netw.*, vol. 9, no. 14, pp. 2557–2575, Sep. 2016.
- [45] M. Bahrololom and M. Khaleghi, "Anomaly intrusion detection system using hierarchical Gaussian mixture model," *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 8, pp. 264–271, 2008.
- [46] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, vol. 382. Hoboken, NJ, USA: Wiley, 2007.
- [47] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J. Roy. Stat. Soc. B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.
- [48] J. G. Campbell, C. Fraley, F. Murtagh, and A. E. Raftery, "Linear flow detection in woven textiles using model-based clustering," *Pattern Recognit. Lett.*, vol. 18, no. 14, pp. 1539–1548, Dec. 1997.
- [49] J. Rissanen, *Stochastic Complexity in Statistical Inquiry Theory*. Singapore: World Scientific, 1989.
- [50] J. J. Oliver, R. A. Baxter, and C. S. Wallace, "Unsupervised learning using MML," in *Proc. 13th Int. Conf. Mach. Learn.*, Jul. 1996, pp. 364–372.
- [51] M. A. T. Figueiredo, J. M. N. Leitão, and A. K. Jain, "On fitting mixture models," in *Energy Minimization Methods in Computer Vision and Pattern Recognition (Lecture Notes in Computer Science)*. Berlin, Germany: Springer, 1999, pp. 54–69.
- [52] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
- [53] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, vol. 454. New York, NY, USA: Springer, 2012.
- [54] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.
- [55] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *Proc. Sci. Inf. Conf.*, Aug. 2014, pp. 372–378.

- [56] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.
- [57] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, Oct. 2006.
- [58] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 5–15, Jan. 2013.
- [59] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. ACM CoNEXT Conf. (CoNEXT)*. New York, NY, USA: ACM, 2008, pp. 1–12.
- [60] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate Internet traffic classification," *Inf. Syst. Frontiers*, vol. 12, no. 2, pp. 149–156, Apr. 2010.
- [61] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *Int. J. Netw. Manage.*, vol. 27, no. 1, p. e1962, Jan. 2017.
- [62] L. Jun, Z. Shunyi, L. Yanqing, and Z. Zailong, "Internet traffic classification using machine learning," in *Proc. 2nd Int. Conf. Commun. Netw. China*, Aug. 2007, pp. 239–243.
- [63] J. Garcia, T. Korhonen, R. Andersson, and F. Vastlund, "Towards video flow classification at a million encrypted flows per second," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2018, pp. 358–365.
- [64] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, Apr. 2011. [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
- [65] D. F. Andrews, "Plots of high-dimensional data," *Biometrics*, vol. 28, no. 1, pp. 125–136, Mar. 1972.
- [66] A. Inselberg, "The plane with parallel coordinates," *Vis. Comput.*, vol. 1, no. 2, pp. 69–91, Aug. 1985.
- [67] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," in *Artificial Intelligence and Statistics*, T. Jaakkola and T. Richardson, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 27–34.



**HASSAN ALIZADEH** received the Ph.D. degree in telecommunications from the University of Aveiro, Aveiro, Portugal. He joined Open University in the Netherlands as a Postdoc Researcher in 2017, where he worked on botnet detection. His current research interests include network security, network intrusion detection, and Internet traffic classification with emphasis on machine/deep learning and data stream mining techniques.



**HARALD VRANKEN** received the M.Sc. degree in information technology, the P.D.Eng. degree in information and communication technology, the Ph.D. degree in electrical engineering, and the M.Sc. degree in science education and communication from the Eindhoven University of Technology, in 1992, 1994, 1998, and 2006, respectively. He was a Senior Scientist with the Philips Research Labs, from 1998 to 2005, where he worked on design-for-testability of digital ICs.

Since 2006, he has been with the Open University in the Netherlands, and since 2014, he has also been with Raboud University. He is currently an Associate Professor. He holds several patents and has coauthored over 50 scientific publications. His current research interests include resilience, software security, network security, and energy analysis of cryptocurrency mining.



**ANDRÉ ZÚQUETE** received the Ph.D. degree in informatics and computer engineering from the Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, in 2001. He is currently an Assistant Professor with the University of Aveiro, Aveiro, Portugal, a Researcher of IEETA (Institute of Electronics and Informatics Engineering of Aveiro), and a Collaborator of the Instituto de Telecomunicações. His research and development activities are centered on the security in distributed

systems, with a focus on the design of security architectures for several specific scenarios (e-Voting, e-Health, e-Government, vehicular networks, etc.). He is a Program Committee Member of several conferences in the areas of security and mobility. He has participated in several national and international projects and did some consulting on the security for Portuguese companies and state Departments. He has dozens of articles published in international forums related with security and mobility and he is the author of a technical book on network security (in Portuguese). He is the Portuguese representative on the IFIP TC11 (Security and Privacy Protection in Information Processing Systems).



**ALI MIRI** is currently a Full Professor with the School of Computer Science, Ryerson University, Toronto. His research interests include cloud computing and big data, computer networks, digital communication, and security and privacy technologies and their applications. He has authored and coauthored more than 200 refereed articles, six books, and eight patents in these fields. He has chaired over a dozen international conference and workshops, and had served on more than 100 technical program committees. Dr. Miri is a member of the Professional Engineers Ontario.

...