Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2016, Article ID 2048302, 21 pages
http://dx.doi.org/10.1155/2016/2048302

Research Article

On Internet Traffic Classification: A Two-Phased Machine Learning Approach

Taimur Bakhshi and Bogdan Ghita

Center for Security, Communications and Network Research, University of Plymouth, Plymouth PL4 8AA, UK

Correspondence should be addressed to Taimur Bakhshi; taimur.bakhshi@plymouth.ac.uk

Received 26 November 2015; Revised 15 April 2016; Accepted 8 May 2016

Academic Editor: Tin-Yu Wu

Traffic classification utilizing flow measurement enables operators to perform essential network management. Flow accounting methods such as NetFlow are, however, considered inadequate for classification requiring additional packet-level information, host behaviour analysis, and specialized hardware limiting their practical adoption. This paper aims to overcome these challenges by proposing two-phased machine learning classification mechanism with NetFlow as input. The individual flow classes are derived per application through *k*-means and are further used to train a C5.0 decision tree classifier. As part of validation, the initial unsupervised phase used flow records of fifteen popular Internet applications that were collected and independently subjected to *k*-means clustering to determine unique flow classes generated per application. The derived flow classes were afterwards used to train and test a supervised C5.0 based decision tree. The resulting classifier reported an average accuracy of 92.37% on approximately 3.4 million test cases increasing to 96.67% with adaptive boosting. The classifier specificity factor which accounted for differentiating content specific from supplementary flows ranged between 98.37% and 99.57%. Furthermore, the computational performance and accuracy of the proposed methodology in comparison with similar machine learning techniques lead us to recommend its extension to other applications in achieving highly granular real-time traffic classification.

1. Introduction

Traffic classification methods using flow and packet based measurements have been previously researched using various techniques ranging from automated machine learning (ML) algorithms to deep packet inspection (DPI) for accurate application identification. Port and protocol analysis, once the default method for traffic identification is now considered obsolete as most applications use dynamic ports, employ HTTPS or encrypted SRTP or use tunnelling, which makes classification close to impossible. Deep packet inspection (DPI) is useful; however the computational overhead and additional hardware required for packet analysis severely limit its practical implementation for network operators [1]. Moreover, aggregation based traffic monitoring techniques using flow measurements have proliferated in recent years due to their inherent scalability and ease of implementation as well as compatibility with existing hardware using standardized export formats such as NetFlow and IPFIX [2]. However, despite an increase in use, flow based network monitoring also encountered traffic classification challenges mainly due to frequent obfuscation and encryption techniques employed by many applications [3–5]. Most automated machine learning classification algorithms utilizing NetFlow involve significant processing overhead and sometimes employ sanitized input requiring simultaneous computations on flow records and packet traces to obtain meaningful results [3, 6, 7]. Additionally, popular Internet applications generate convoluted sets of flows representing content specific and auxiliary control flows, making application identification on a per-flow basis even more challenging. The present paper proposed a per-flow C5.0 decision tree classifier by employing a two-phased machine learning approach while solely utilizing the existing quantitative attributes of NetFlow records. Flow records for fifteen popular Internet applications were first collected and unique flow classes were derived per application using *k*-means clustering. Based on these

Table 1: Traffic classification approaches.

Category	Classification methodology	Attribute(s) used	Granularity	Processing time	Sample tools/ML techniques
Port based	Protocol port	Protocol ports	High	Low	Any (custom), PRTG network monitor [55], Nagios [56], Wireshark [48]
Payload inspection	Deep packet inspection	Payload inspection of, for example, first n packets, first packet per direction	High	High	OpenDPI [1], nDPI [45], L7 (TIE) [35]
	Stochastic packet inference	Statistical properties inherent in packet header and payload	High	High	Netzob [57], Polyglot [58], KISS [8]
Behavioural techniques	End-point behaviour monitoring	Identifying host (communication) behaviour pattern	Low	Moderate	BLINC [46], SVM [59], naïve Bayes [60]
	Traffic accounting	Heuristic analysis of inspected packets, flows	High	High	ANTCs [61], naïve Bayes [60], Bayesian network [62]
Statistical approaches	Packet based	Packet and payload size, interpacket arrival time	High	Moderate	kNN [63], Hidden Markov/Gaussian Mixture Models
	Flow based	Duration, transmission rate, multiple flow features	Low	Low	k-means/hierarchical clustering [27], J48 [30], C5.0 [31], BFTree [64], SVM [59]

preclassified flows (the ground-truth data), the C.50 classifier was subsequently trained for highly granular per-flow application traffic classification. The classified applications included YouTube, Netflix, Dailymotion, Skype, Google Talk, Facebook video chat, VUZE and BitTorrent clients, Dropbox, Google Drive and OneDrive cloud storage, two interactive online games, and the Thunderbird and Outlook email clients. The rest of this paper is organized as follows. Section 2 presents related background work in traffic classification and gives an overview of k-means clustering and C.50 algorithm. Section 3 elaborates on data collection, preprocessing, and feature selection methodology. Section 4 details flow clustering using k-means and discusses the derived flow classes. Section 5 evaluates the accuracy of the resulting C5.0 classifier while Section 6 compares the performance and computation overhead of the proposed approach with state-of-the-art ML based classification schemes. Final conclusions are presented in Section 7.

2. Background

The following subsections present a comprehensive overview of state of the art in traffic classification and consider related work in addressing flow level classification challenges using supervised, unsupervised, and cascaded ML techniques. A brief outline of k-means clustering and C5.0 machine learning techniques in the context of traffic classification is detailed afterwards.

2.1. Traffic Classification Methodologies and Related Work. Traffic classification serves as a fundamental requirement for network operators to differentiate and prioritize traffic for a number of purposes, from guaranteeing quality of service to anomaly detection and even profiling user resource requirements. Consequentially a large body of research can be attributed to traffic classification such as [8–13] along with comprehensive surveys [14–16], which reflect the interest of the networking community in this particular avenue. From a high level methodology perspective, traffic classification research can be broadly divided into port and packet payload based classification, behavioural identification techniques, and statistical measurement based approaches [16]. A summary of the prevalent classification approaches, their traffic feature usage, and associated algorithms is given in Table 1. While port based classification techniques are now considered obsolete given the frequent obfuscation techniques and dynamic range of ports used by applications, packet payload inspection methods remain relevant primarily due to their high classification accuracy. Payload based classifiers inspect packet payloads using deep packet inspection (DPI) to identify application signatures or utilize a stochastic inspection (SPI) of packets to look for statistical parameters in packet payloads. Although the resulting classification is highly accurate it also presents significant computational costs [16–18] as well as being error-prone in dealing with encrypted packets. In comparison, behavioural classification techniques work higher up the networking stack and peruse the total traffic patterns of the end-points (hosts and servers) such as the number of machines contacted, the protocol used, and the time frame of bidirectional communication to identify the application being used on the host [19–22]. Behavioural techniques are highly promising and provide a great deal of classification accuracy with reduced overhead compared to payload inspection methods [9, 13]. However, behavioural techniques focus on end-point activity and require parameters from a number of flows to be collected and analysed before successful application identification. With increasing ubiquity of flow level network monitoring which presents a low-cost traffic accounting solution, specifically utilizing NetFlow due to scalability and ease of use, statistical classification

techniques utilizing flow measurements have gained momentum [2, 8–12, 23]. Statistical approaches exploit application diversity and inherent traffic footprints (flow parameters) to characterize traffic and subsequently derive classification benchmarks through data mining techniques to identify individual applications [24]. Statistical classification is considered light-weight and highly scalable from an operational point of view especially when real-time or near real-time traffic identification is required. While traffic classification in the network core is increasingly challenging and seldom implemented, application flow identification at the edge or network ingress as detailed in [16] allows operators in shaping the respective traffic further upstream. Statistical flow based traffic classifications however, due to minimal number of available features in a typical flow record such as NetFlow, report low classification accuracy and increasingly rely on additional packet payload information to produce effective results [8–12]. The present work picks up from this narrative and solely utilizes NetFlow attributes using two-phased machine learning (ML), incorporating a combination of unsupervised k-means cluster analysis and C5.0 based decision tree algorithm to achieve high accuracy in application traffic classification.

Typical statistical flow level classification can be further subdivided based on the type of ML algorithm being used, that is, supervised or unsupervised. Unsupervised methods alone do not rely on any training data for classification and, while being time and resource efficient, especially with large data sets, encompass significant limitations hampering their wider adoption. Firstly, cluster analysis is mostly done offline and relies on evaluating stored flow records in statistical applications for cluster learning and traffic identification [25, 26]. Secondly, unsupervised clustering quite often also requires additional information from packet-level traces requiring specialized hardware and is therefore considered as an expensive option for network operators [27, 28]. Lastly, once traffic records have been clustered, defining optimal value ranges of classification attributes for real-time systems is seldom easy and highly dependent on the data set used [29].

Supervised ML algorithms in contrast require a comprehensive training data set to serve as primary input for building the classifiers; the completeness of the data set, together with the ability of the method to discriminate between classes, is the decisive factor for the accuracy of the method. Although considered favourable in terms of presenting a discrete rule set or decision tree for identifying applications, supervised training also falls short of presenting a complete solution to classification challenges, as a highly accurate training/test data set (also referred to as ground-truth data) is required prior to further use. To aid in obtaining accurate ground-truth data several ideas have been explored. Separate offline traffic identification systems were used to preprocess and generate training data for online classifiers in [30]. Custom scripts were employed in [31] on researcher machines to associate flow records and packets with application usage. Deep packet inspection was used to obtain application names for labelling training data in [32]. However, obtaining accurate ground-truth data considering only singular application class labels for subsequent training of the supervised ML classifier falls significantly short of recognizing the different flows generated per application [25–32]. Internet applications generate a convoluted set of flows including both application initiated content specific or auxiliary control flows and other functional traffic such as DNS or multicasts. Per-flow traffic classification hence requires a full appreciation of the peculiar traits and types of flows (classes) generated per application to eliminate the classification system relying on time window analysis or packet derivative information to achieve higher classification accuracy.

To increase the flow classification accuracy, cascaded classification methodologies employing a combination of algorithms as well as semisupervised ML approaches have also been previously explored. Foremski et al. [33] combined several algorithms using a cascaded principle where the selection of chosen algorithm to be applied for each IP flow classification depended on predetermined classifier selection criteria. Jin et al. [23] combined binary classifiers in a series to identify traffic flows while using a scoring system to assign each flow to a traffic class. Additionally, collective traffic statistics from multiple flows were used to achieve greater classification accuracy. Similarly Carela-Español et al. [34] used k-dimensional trees to implement an online real-time classifier using only initial packets from flows and destination port numbers for classification. de Donato et al. [35] introduced a comprehensive traffic identification engine (TIE) incorporating several modular classifier plug-ins, using the available input traffic features to select the classifier(s), merging the obtained results from each, and giving the final classification output. A similar approach was followed in Netramark [36] incorporating multiple classifiers to appraise the comparative accuracy of the algorithms as well as use a weighted voting framework to select a single best classification output. Another prominent ML tool used in traffic classification studies is Weka [37], incorporating a library (Java based) of supervised and unsupervised classifiers which can be readily implemented on test data set to evaluate the accuracy of the results from each methodology. Using multiple classifiers and selecting the best choice for classifying each traffic flow through voting or even combining the results for a final verdict, however, does not specifically consider refining the ground-truth data to fully account for the multiple flow classes (per application) and their subsequent identification. Additionally merging multiple instances of classifiers raises scalability issues with regard to their real-time implementation.

Semisupervised learning techniques on the other hand use a relatively small amount of labelled data with a large amount of unlabelled records to train a classifier [38]. Two ML algorithms, unsupervised and supervised, were combined in [39] and the scheme used a probabilistic assignment during unsupervised cluster analysis to associated clusters with traffic labels. Zhang et al. [40] proposed using a fractional amount of flows labelled through cluster analysis to train and construct a classification model specifically focusing on zero-day application identification. The sole use of cluster analysis to serve as a means for identifying applications and generating training data without either additional manual or automated validation may, however, lead to incorrect traffic labelling. Unmapped flow clusters

from unsupervised learning were, for example, attributed to unknown traffic in [39]. Error-prone labelling of flows through cluster analysis using semisupervised approaches may also result in significant misclassification penalties.

Subflow qualification is paramount to fully apply network policies such as guaranteeing application QoS, profiling user activity, and accurately detecting network anomalies. Furthermore, correct subflow identification aids in reducing the over-time degradation of supervised algorithms by accounting for the multiple types of flow classes and their respective parameters per application, reducing the unseen examples. The present approach refines the acquired ground-truth data by segregation of prelabelled application flows through independent unsupervised clustering, thereafter used to train a supervised C5.0 decision tree. The resulting classifier is hence able to recognize the multiple flow classes even from the same application without combining the results from multiple classifiers or using popular voting. This also increases the scalability of the final decision tree which can be implemented as a standalone system at suitable traffic aggregation points in the network capable of real-time traffic classification.

Finally, as noted in [25, 41, 42], given the multitude of classification methodologies, dissimilar traffic traces, and the diversity in flow classification features, benchmarking the performance of classification algorithms is a difficult undertaking. In the present work, the widely used classification tool Weka [36] was employed to yield a qualitative comparison in terms of the accuracy and computational overhead of the proposed design against some state-of-the-art classification methods.

2.2. k-Means Clustering. Flow level clustering requires efficiently partitioning collected flows per application into groups based on exported NetFlow attributes. One of the prominent unsupervised clustering techniques is the k-means clustering algorithm preferred over other methods such as hierarchical clustering, due to its enhanced computational efficiency [10, 32]. k-means minimizes a given number of vectors by choosing k random vectors as initial cluster centres and assigning each vector to a cluster as determined by a distance metric comparison with the cluster centre (a squared error function) as given in (1). Cluster centres are then recomputed as the average (or mean) of the cluster members. This iteration continues repeatedly, ending either when the clusters converge or when a specified number of iterations have passed [27, 43]:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2. \quad (1)$$

In (1), c_j represents cluster centre, n equals the size of the sample space (collected flows), and k is the chosen value for number of unique clusters (flow classes). Hence, using k-means, n flows can be partitioned into k classes. Value of k is of significant importance as it directly influences the number of flow classes affecting overfitting. An intelligent alternative to calculate the optimal number of clusters is by using the Everitt and Hothorn graphical approach [44], discussed and applied in Section 4.

2.3. C5.0 Machine Learning Algorithm. The C5.0 algorithm and its predecessor C4.5 described in [30] attempt to predict a dependent attribute by finding optimal value ranges of an independent set of attributes. At each stage of iteration, the algorithm aims to minimize information entropy by finding a single attribute that best separates different classes from each other. The process continues until the whole sample space is split into a decision tree isolating each class. Hence, in a sample space comprising n application flow classes, if training data is given by preclassified samples given by vector S (2), each sample flow f_n may consist of a j-dimensional vector (3), where z_j represents independent attributes which are used to identify the class in which f_n falls:

$$S = [f_1, f_2, f_3, f_n] \quad (2)$$

$$f_n = \left[z_1, z_2, z_3, z_j\right]. \quad (3)$$

C5.0 could therefore be used to build a decision tree utilizing flow attributes z_j of each sample f_n from preclassified training data. C5.0 also includes advanced options for boosting, pruning, and winnowing to enhance accuracy and computational efficiency of the resulting decision tree classifier [31]. The adaptive boosting proposed in [32] generates a batch of classifiers instead of a single classifier and uses vote count from each classifier on every examined sample to predict the final class. Advanced pruning options remove parts of the classification tree representing relatively high error rate at every stage of iteration and once finally for the complete tree to reduce performance caveats. Finally enabling winnowing reduces the feature set required for classification by removing covariates with low predictive ability during classifier training and cross-validation stage.

3. Methodology

To address the challenges of obtaining high quality ground-truth data incorporating flow class segregation and identification in each of the examined applications, our proposed classification technique utilizes unsupervised cluster analysis and supervised classifier training in tandem. A high level overview of the traffic classification scheme is shown in Figure 1 with a description of principal steps as follows.

(i) *Preprocessing*. Internet traffic is collected from end-user machines and marked with application labels accordingly (e.g., Skype and YouTube) using a localized operational packet-level classifier. Application labelled traffic is afterwards exported as flows using a flow exporting utility for unsupervised cluster analysis.

(ii) *Cluster Analysis*. Using unsupervised k-means, flows belonging to individual applications are separately cluster analysed to extract unique subclasses per application, offering a finer granularity of the classification (e.g., YouTube and Netflix flows would be classed as streaming and browsing).

(iii) *Classifier Training*. Flows marked with their k-means clusters, indicating the subclass they belong to, are

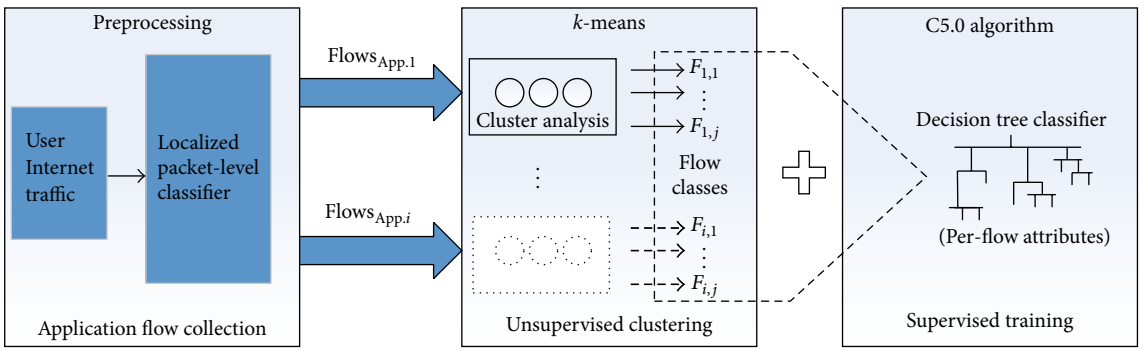


Figure 1: Traffic classification scheme.

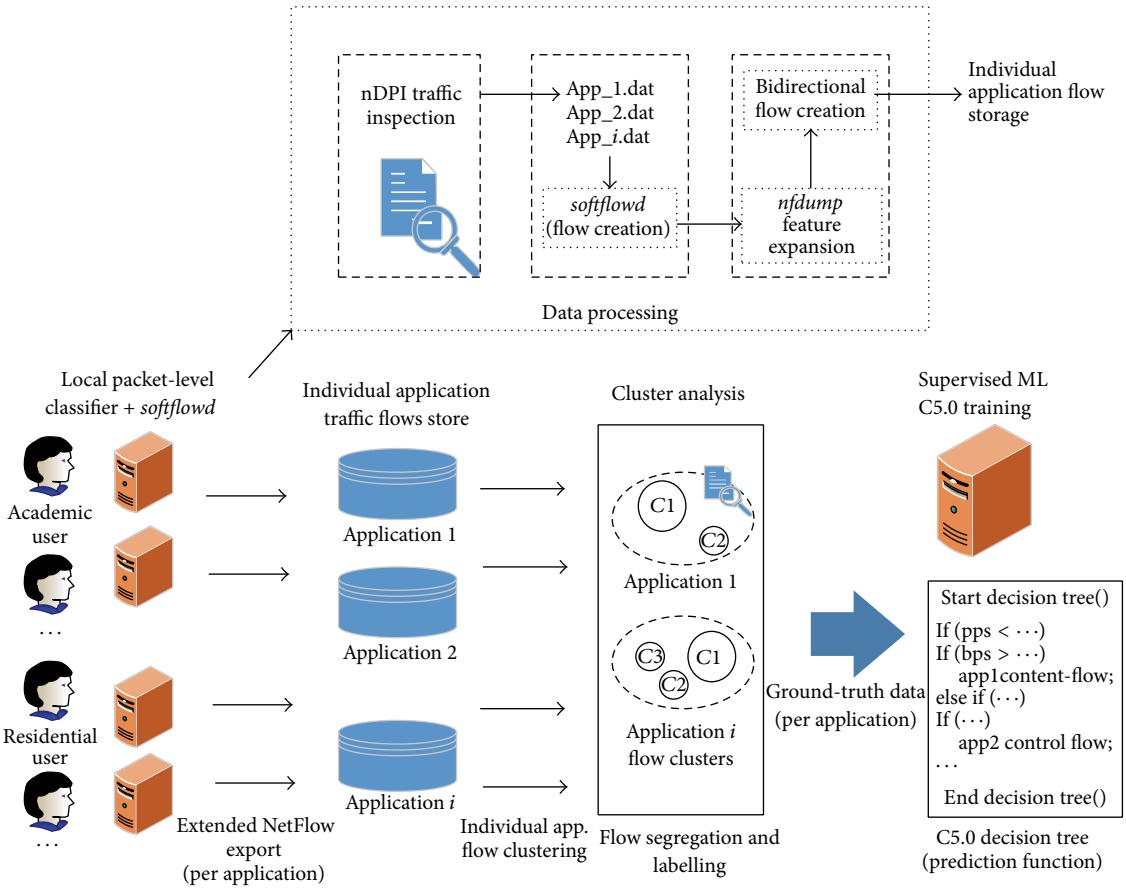


Figure 2: Data collection and preprocessing workflow.

afterwards fed to a C5.0 classifier for supervised training, leading to a decision tree.

(iv) *Evaluation*. A separate data set is used for testing the accuracy of the algorithm. For each NetFlow record the trained C5.0 classifies the application and the subclass of the flow based on their respective attributes, ingrained during decision tree creation.

The following subsections detail the methodology used for collecting NetFlow records from user machines, flow customization, k-means clustering, and designing feature sets for the C5.0 classifier.

3.1. Data Collection. To increase the scalability of the resultant classifier in identifying traffic from different network settings, NetFlow records were collected from two environments: (i) typical residential premises using broadband connection and (ii) an academic setting using corporate Internet as depicted in Figure 2. Two PCs were used in each environment for user traffic generation and collection. In order to accurately isolate traffic for each of the fifteen examined applications, a localized extension of packet-level classifier nDPI [45] was used on the researcher's machines, excluding references to application data or the end-point identity of users for anonymity similar to [46, 47]. The nDPI is based on the

Table 2: Traffic collection summary.

Traffic class	Application	Bytes ($\times 10^6$)	Flows	Dates	Duration (hrs)
Video streaming	YouTube	16093.87	879641	[09–12]/09/2015	6.89
	Netflix	11586.61	454985	[08–09]/09/2015	5.65
	Dailymotion	11258.12	398412	[15–16]/03/2016	5.31
Video chat/VoIP	Skype	6251.06	1492380	[08–17]/10/2015	9.45
	Gtalk	4584.02	1025260	[14–18]/03/2016	4.25
	Facebook Messenger	7824.13	1158302	[15–21]/03/2016	3.28
P2P torrent	VUZE	131611.31	1318749	[20–23]/09/2015	4.28
	BitTorrent	154138.97	1308881	[20–23]/09/2015	3.56
Cloud storage	Dropbox	211833.57	408677	[11–23]/09/2015	1.56
	Google Drive	158923.52	358426	[20–23]/03/2016	2.31
	OneDrive	186358.21	325854	[21–27]/03/2016	1.81
Online games	8-Ball Pool	953.91	1358425	[10–13]/10/2015	0.35
	Treasure Hunt	1158.28	1592362	[15–22]/03/2016	2.11
Email client	Thunderbird	1401.36	821484	[15–31]/08/2015	2.21
	Outlook	1854.54	698722	[19–31]/03/2016	3.55

libcap and OpenDPI library [48] and is continuously updated to increase the number of applications and protocols that can be successfully identified. Once the traffic from the examined applications was identified and marked with application names, it was converted to the NetFlow format using the *softflowd* utility [49]. A total of approximately 13.6×10^6 flows were collected and marked with application labels. Table 2 presents a summary of collected flows including the bytes, flows, time frame of the traffic collection, and the duration associated with each application. The NetFlow records were afterwards subjected to further preprocessing, that is, feature set expansion using the nfdump utility [50] and creation of bidirectional flows before being exported to individual application storage filers as detailed in the following section.

3.2. Customizing NetFlow Records. NetFlow by default outputs 5-tuple address, port, and protocol connection information ⟨SrcIP, DstIP, SrcPo, DstPo., Proto.⟩ along with the timing and interface relating to each flow. Transmitted and received flows are, however, not correlated by default. Generally considered as lacking an extensive set of attributes, it further extrapolates the use of packet traces for traffic identification as highlighted in [26–29]. To fully explore the prediction ability of NetFlow attributes with the proposed methodology, *nfdump* [50] was used to expand the NetFlow output to display flow duration, number of packets, data rate (bits per second), packet transfer rate (pps), and bytes per packet (Bpp) for each flow; then transmitted and received flows were correlated to output a 17-tuple bidirectional flow as shown by the snippet in Table 3.

3.3. Extracting Flow Classes (k-Means Clustering). Popular applications such as YouTube or Skype generate an intricate set of flows between various web servers and the client depending on their underlying content distribution, load balancing, and authentication schemes [51–54]. While DPI based traffic classification is useful in identifying the respective applications, it does not specifically segregate different flows generated per application attributed to the primary application content or control signalling, session establishment, embedded webpage advertisements, and so forth. Per-flow classification consequently requires a separation of content specific and supplementary flows to retrieve the different flow classes generated per application for subsequently training and testing the classifier. Flow classification is not possible using supervised ML alone due to lack of information about the flow classes generated by an application, requiring an independent technique for per-application flow segregation. The k-means algorithm was therefore independently applied on paired bidirectional flows generated per application in order to retrieve the respective flow classes. Due to extensive repetition of source and destination IP addresses, port numbers, and protocol information in the collected data, these were deemed scalar entities for analysis and excluded while clustering. The remaining 12 attributes chosen to isolate application specific flows from auxiliary data per application for further analysis comprise transmitted bytes Tx.B., transmitted packets Tx.Pkt., transmitted data rate in bits per second Tx.bps., transmitted packers per second Tx.pps., transmitted packet size in bytes per packet Tx.Bpp., transmitted flow duration Tx.s., received bytes Rx.B., received packets Rx.Pkt., received data rate in bits per second Rx.bps., received packets per second Rx.pps., received packet size in bytes per packet Rx.Bpp., and received flow duration Rx.s. The clustering vector per application could therefore be represented by the following equation:

$$F_{ij} = \big[\mathrm{Tx.B}_{ij}, \mathrm{Tx.pkt}_{ij}, \mathrm{Tx.bps}_{ij}, \mathrm{Tx.pps}_{ij}, \mathrm{Tx.Bpp}_{ij}, \mathrm{Rx.s}_{ij}, \mathrm{Rx.B}_{ij}, \mathrm{Rx.pkt}_{ij}, \mathrm{Rx.bps}_{ij}, \mathrm{Rx.pps}_{ij}, \mathrm{Rx.Bpp}_{ij}, \mathrm{Rx.s}_{ij}\big]. \quad (4)$$

In (4), i and j are unique per application and per flow, respectively. Hence, bidirectional flows represented by vector F_{ij} split into k clusters represent the types of flows per

Table 3: 17-tuple bidirectional NetFlow records.

	SrcIP	DstIP	Prot.	SrcPo	DstPo	Tx.B.	Tx.Pkt.	Tx.s.	Tx.bps.	Tx.pps.	Tx.Bpp.	Rx.B.	Rx.Pkt.	Rx.s.	Rx.bps.	Rx.pps.	Rx.Bpp.
12	*Private IP* address	Website/application	TCP	59648	80	1737	10	16.551	839	0	173	2768	10	16.542	1338.0	0	276
13	*Private IP* address	Website/application	TCP	50254	443	763	8	0.073	83616	109	95	4571	7	0.063	580444.0	111	653
14	*Private IP* address	Website/application	TCP	37832	443	397	5	0.078	40717	64	79	3657	4	0.041	713560.0	97	914
15	*Private IP* address	Website/application	TCP	47216	443	2663	12	1.291	16501	9	221	5718	11	1.279	35765.0	8	519
16	*Private IP* address	Website/application	TCP	53509	443	883	10	0.278	25410	35	88	4472	9	0.243	147226.0	37	496

Table 4: NetFlow feature sets for C5.0 classifier training.

Set 1	Set 2
Protocol and port information (i) Source and destination port *num* (ii) Protocol (TCP, UDP)	Protocol and port information (i) Source and destination port *labels* (ii) Protocol (TCP, UDP)
Set 3	**Set 4**
Flow parameters (i) Received and transmitted packets (Rx.Pkts., Tx.Pkts.) (ii) Received and transmitted packet rate (Rx.pps., Tx.pps.) (iii) Received and transmitted data rate (Rx.bps., Tx.bps.) (iv) Received and transmitted bytes per packet (Tx.Bpp., Rx.Bpp.) (v) Received and transmitted data (Rx.B., Tx.B.) (vi) Received and transmitted flow duration (Tx.s., Rx.s.)	Flow parameter ratios (i) Received packets to transmitted packets (Rx.Pkts./Tx.Pkts.) (ii) Received to transmitted packet rate (Rx.pps./Tx.pps.) (iii) Received to transmitted data rate (Rx.bps./Tx.bps.) (iv) Received to transmitted bytes per packet (Rx.Bpp./Tx.Bpp.) (v) Received to transmitted data (Rx.B./Tx.B.) (vi) Received to transmitted flow duration (Rx.s./Tx.s.)

application. Once segregated, flows per application were subsequently labelled with the respective flow class before data sets for all the fifteen examined applications were combined and split in equal proportions (~50%) for training and testing the C5.0 ML classifier.

3.4. Feature Selection. Feature set selection is of paramount importance for training the classifier, given that these should be predictive and must correctly classify the application traffic. The selected features must also closely link to the flow classes derived from k-means clustering and utilize their NetFlow values to discriminate between different application flows. NetFlow attributes can be broadly grouped by transport layer parameters and network layer traffic statistics for each flow. Both groups were studied for classifier training individually and in combination to examine their efficiency for classification. Additionally, minimizing the set of features for traffic classification also minimizes the processing overhead involved in creating decision trees and reduced classification time. Four sets of features sets were, therefore, devised around transport and network layer features translating for the independent attributes z_j, given in (3) as shown in Table 4. Set 1 included source and destination port numbers along with protocol information. Set 2 used source and destination ports however, rather than using actual port numbers; these were labelled as Known (0–1023) and Unknown (>1023) aiming to evaluate classification accuracy on basic port information alone. Set 3 included 12 flow attributes excluding source and destination IP addresses and port and protocol information while set 4 represented the same as ratios thereby reducing the feature set to 6 covariates with the intention of compressing the size of resulting decision tree even further.

4. Unsupervised Flow Clustering

4.1. Calculating Flow Classes per Application: Value of k. A total of 6.8 million bidirectional flows were cluster analysed independently for each application using the computationally efficient Hartigan and Wong implementation of k-means in *R* [43]. Since value of k directly influences the number of flow clusters (classes) per application, Everitt and Hothorn method was employed to determine k number per application [44]. This graphical technique plots *within cluster sum of square values* (wss) against the number of clusters k, with the curve in plot signifying an appropriate number of clusters that fit the input data. The plot of wss versus k of flow records for each application is given in Figures 3–7. Automated scripting calculated the maximum *within cluster variance* between successive values according to Everitt and Hothorn criteria in reaching the optimal cluster number per application and marked the respective flow records with the individual cluster colour. Table 5 details the optimal number of clusters translating for different types of flows classes determined per application along with the "within sum of squares" per cluster to "total sum of square distance" between clusters (wss/total_ss) representing the tightness of these clusters in covering the entire sample space, that is, flow records. A small sample set comprising approximately 1K bidirectional flows from each cluster was afterwards analysed offline to assign the respective flow labels as detailed in the following section.

4.2. Analysis. YouTube access seemed to be solely used for *streaming* (and not content upload) in the present case and the corresponding clusters indicated 3 unique flow classes generated as shown by the graph in Figure 3(a). According to YouTube traffic analysis studies carried out in [51, 52], these were narrowed to three unique flow classes and attributed to content-streaming, website browsing (or video searches), and redirections between YouTube and other Google content distribution servers. Netflix and Dailymotion video streaming similarly showed three flow classes, two for video content-streaming having different download rates corresponding to start of video succeeded by steady buffering stage and a third for user searches. For these applications, video streaming flows were labelled as "streaming" while website searches and server redirections were labelled as "browsing."

The Skype client was used for *video with voice communication* rather than file sharing or instant messaging as per the labelled flow record perusal. Subsequent clustering produced two highly discriminate clusters given by the knee-point of the graph in Figure 4(a). Skype stores user information in a decentralized manner with Skype clients acting as

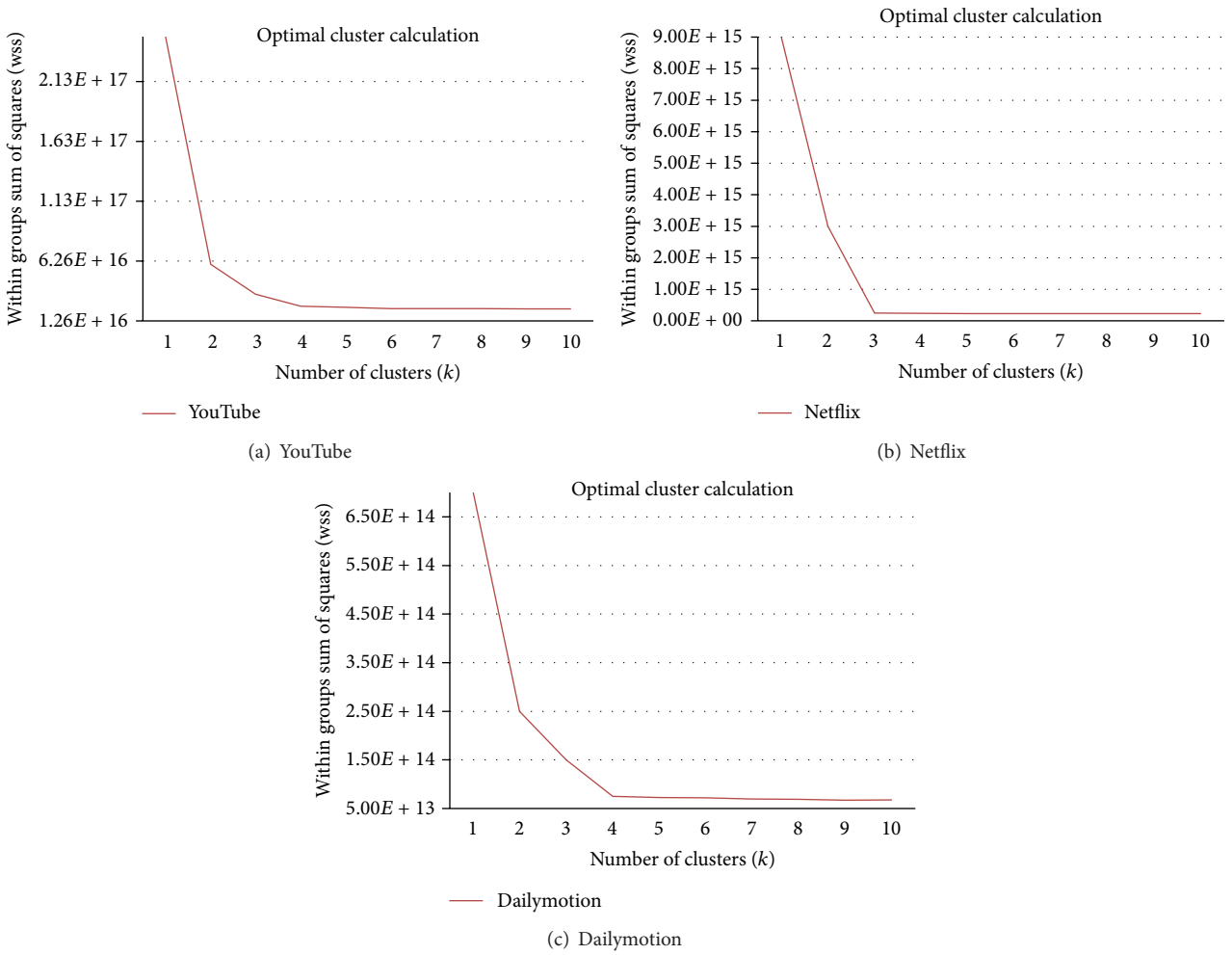


Figure 3: Inner-cluster variance versus k.

Table 5: Segregated flows per application.

Traffic class	Application	Cluster (k)	wss/total_ss	Content specific flows	Auxiliary flows
Streaming	YouTube	3	87.3%	Streaming	Browsing
	Netflix	3	94.6%	Streaming	Browsing
	Dailymotion	3	95.1%	Streaming	Browsing
Comms./VoIP	Skype	2	98.8%	Comms.	Comms. Ctrl.
	Gtalk	2	97.21%	Comms.	Comms. Ctrl.
	Facebook Messenger	3	92.12%	Comms.	Comms. Ctrl., Browsing
Torrents/P2P	VUZE	3	97.9%	Torrent	Torr.Ctrl.
	BitTorrent	3	91.2%	Torrent	Torr.Ctrl.
Cloud storage	Dropbox	3	89.2%	Up/dwnld.	Browsing
	Google Drive	3	88.15%	Up/dwnld.	Browsing
	OneDrive	3	92.14%	Up/dwnld.	Browsing
Gaming	8-Ball Pool	2	88.4%	Game ctrl.	Game setup
	Treasure Hunt	2	91.98%	Game ctrl.	Game setup
Email	Thunderbird	2	99.14%	Email msg.	Dir. lookups
	Outlook	2	97.45%	Email msg.	Dir. lookups

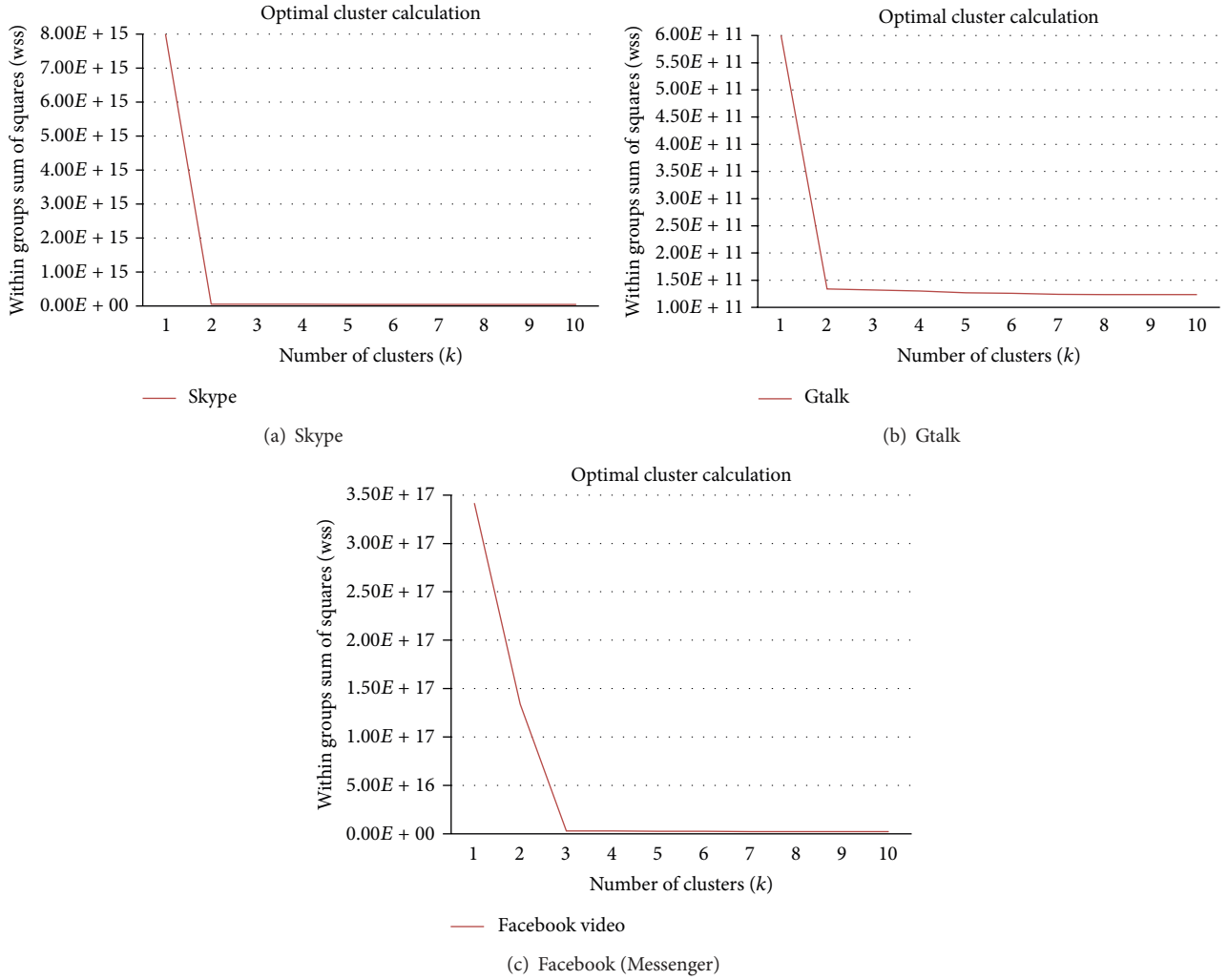
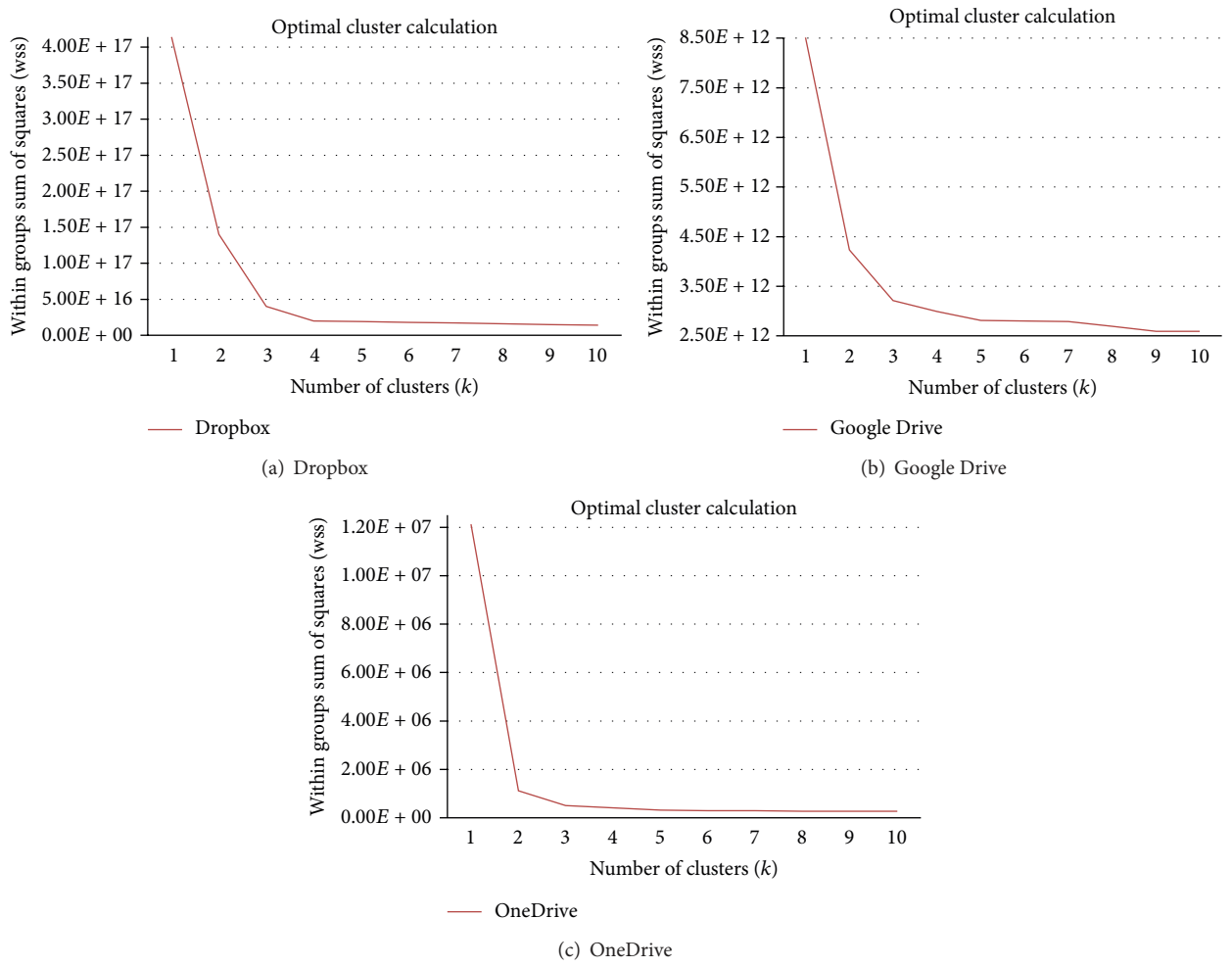


Figure 4: Inner-cluster variance versus k.

host nodes that initiate connections with super nodes for registering with a Skype login server and exchanging continuous keep-alive messages [53]. The resulting overlay peer-to-peer network employs both TCP and UDP connections both for communication between host and super nodes and for communication between two hosts running the client application [54, 65]. One flow cluster was hence determined to be directly associated with control features servicing connections and authentication between host and super nodes, having a much lower data volume and receiving rate and a significant number of unidirectional flows compared to the second group. The second flow cluster is comprised of video calls between Skype clients having substantially higher data rate and total data volume. The respective flows were labelled as "Comms. Control" and "Comms." accordingly. The same number of clusters was observed for Gtalk attributed to voice communication and control signalling with the Google content server with the later having a lower traffic footprint with respect to flow transmission duration and the average bit rate of the flows compared to the former. For Facebook Messenger, however, three optimal clusters were observed, one with a high bit rate and duration similar to the VoIP calls observed in Skype and Gtalk, one for connection establishment, and lastly one for the background live newsfeed being continuously updated on the Facebook page. The clusters were thus accordingly labelled under "Comms." and "Comms. Control" and "Browsing" classes.

For *online cloud storage*, usually requiring low user interactivity as highlighted in [66], the prominent Dropbox storage, Google Drive, and OneDrive were examined. The applications employed file transfers ranging in size from 25 KB to 1.5 GB, frequently in batches of 1, 5, and 10 files. Cluster analysis on generated traffic featured around 3 optimal flow clusters as represented by Figure 5. The three distinct flow clusters after analysis were labelled as one each for file "uploads" and "downloads" and a third for interaction with the hosting website tagged "browsing."

To examine *torrent applications*, the original BitTorrent and VUZE derivative client were used on researcher machines to search and download different combinations of files with sizes ranging from 25 MB to over 1 GB. Cluster analysing these torrent flows resulted in three distinct clusters



(a) Dropbox

(b) Google Drive

(c) OneDrive

Figure 5: Inner-cluster variance versus k.

representing actual file download labelled as "torrent" and later two as "torrent control" responsible for further seeding of downloaded files and communication with other peers.

For online interactive Macromedia Flash player based pool and Treasure Hunt game, two clearly distinct flow classes as depicted in Figures 6(c) and 7(a) responsible for initial "game setup" and continued interactive "game control" constituted all flows.

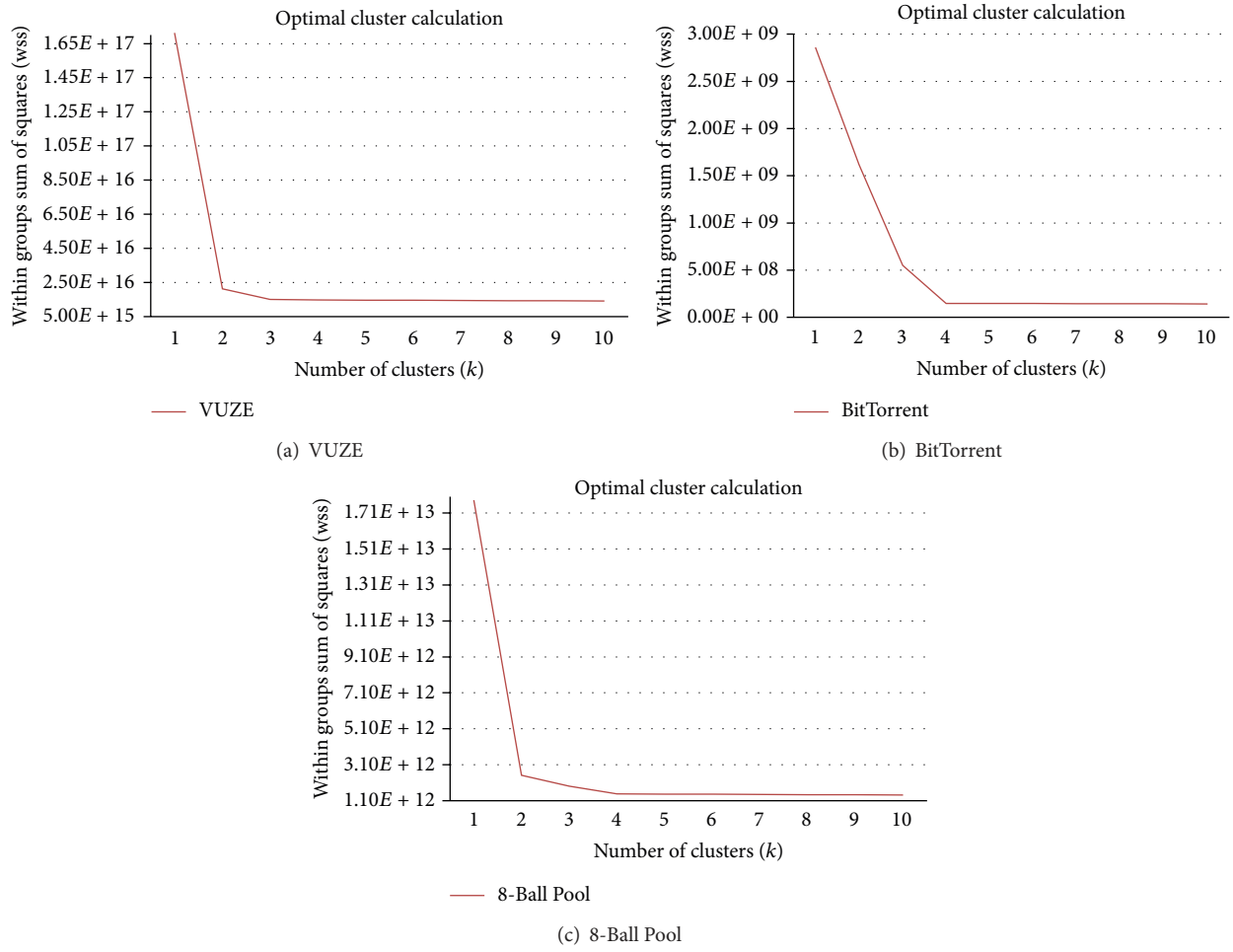
Lastly the email clients Thunderbird and Outlook were used with three distinct email accounts, Yahoo, Gmail, and a corporate account. Cluster analysis revealed two discrete types of flows shown in Figures 7(a) and 7(b). One flow cluster comprised sending and receiving email messages which in this case could also be easily identified by looking at well-known destination port assignments for SMTP, POP, and IMAP protocols. The second flow class represented "directory lookups" by the client using HTTP and SSL having significantly lower total data volume per flow compared to email messages.

Segregated flows of all applications were labelled with flow classes and combined into a single data set. The next section details the splitting of training and testing data and evaluates the C5.0 ML classifier.

5. C5.0 Decision Tree Classifier

Approximately 6.8 million flows were labelled with appropriate flow classes as a result of k-means cluster analysis, in accordance with Table 4. In order to comprehensively test classifier accuracy, the data set was further split in almost equal percentages (~50%) per flow class for training and testing purposes.

5.1. Classifier Evaluation. C5.0 ML was applied on the training data set using feature sets 1 to 4, with alternate pruning and boosting options. As mentioned earlier, enabling pruning removes parts of the decision tree representing relatively higher error rates than others while adaptive boosting generates a batch of classifiers and uses voting on every examined sample to predict the final class. Classifiers were derived by enabling both options to analyse improvements in predictive ability using the feature sets in Table 4. The resulting prediction accuracy for each attribute set is reported in Table 6. Set 1 included source and destination port numbers along with protocol information and resulted in a maximum accuracy of 41.97% with the maximum allowed boosting factor of 100 and could easily be ruled out for use as standalone feature set for



(a) VUZE

(b) BitTorrent

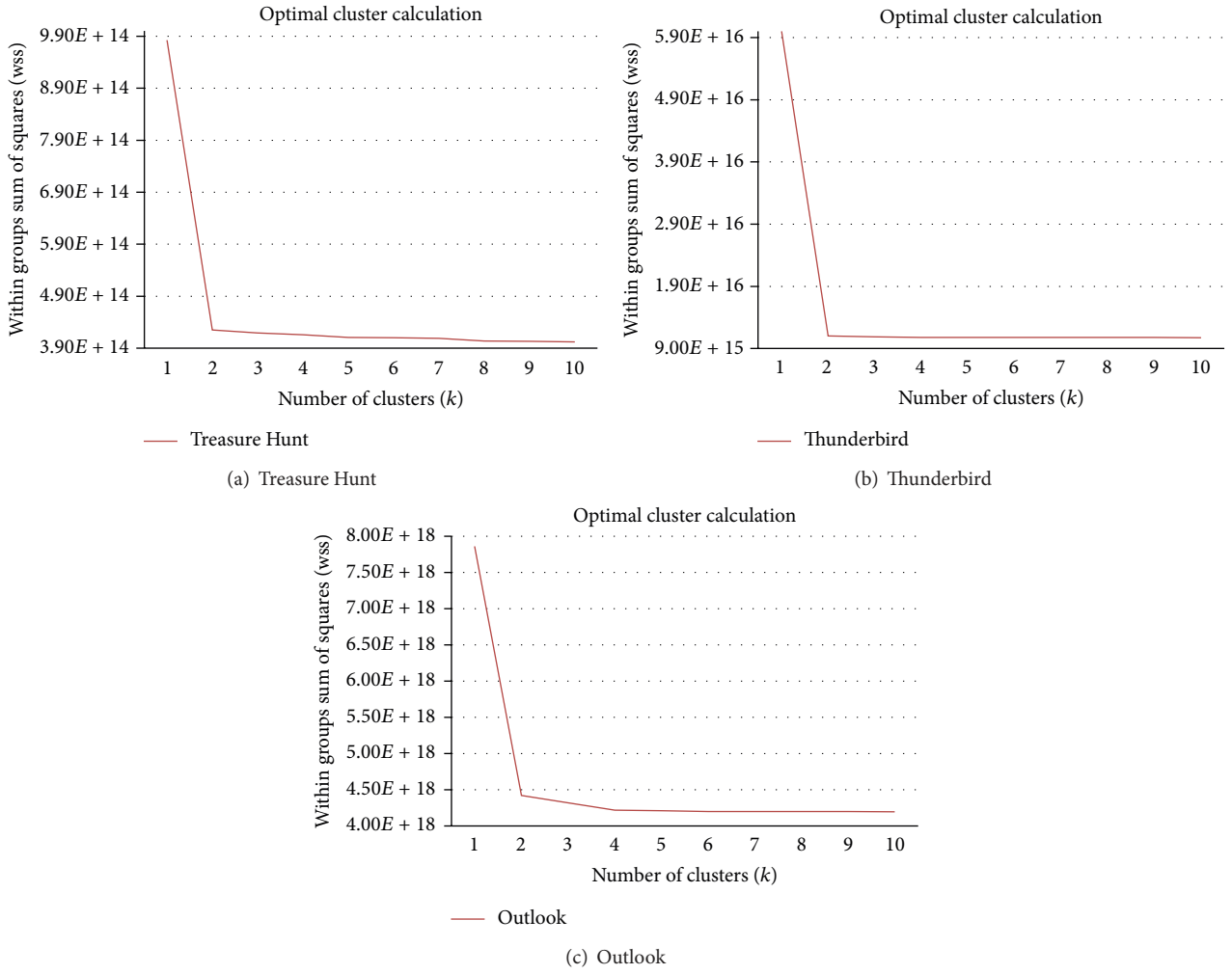
(c) 8-Ball Pool

Figure 6: Inner-cluster variance versus k.

Table 6: Feature sets versus classifier accuracy.

Feature set	Pruning = false			Pruning = true		
	No boost	Boost 10	Boost 100	No boost	Boost 10	Boost 100
Set 1	39.58	40.01	41.34	39.44	40.48	41.97
Set 2	24.29	24.29	24.29	24.29	24.29	24.29
Set 3	82.29	83.24	84.29	82.20	84.97	83.95
Set 4	73.18	75.51	75.70	73.18	72.62	75.03
Sets 1 + 3	91.37	94.39	95.98	92.37	94.52	96.67
Sets 1 + 4	84.48	87.47	86.47	84.48	86.42	86.79
Sets 2 + 3	84.90	86.91	85.71	84.90	85.00	85.61
Sets 2 + 4	74.37	77.07	77.21	74.37	76.83	77.42

classification. Set 2 used port name labelling instead of actual numbers and protocol information, resulting in considerably low accuracy even when compared to set 1 with uniformity in values regardless of boosting at 24.29%. Set 3 included twelve flow attributes and resulted in a significantly improved accuracy of 84.97% with a boost 10. Finally, set 4 incorporating only six flow ratios led to a maximum accuracy of 75.03% with 100 times' boost. In this particular instance disabling pruning resulted in a more accurate classifier at 75.70%. When used in combinations sets 2 and 4 presented lowest accuracy peaking at 77.42% while sets 1 and 4 as well as 2 and 3 resulted in reasonable level of classifier accuracy at 86.79% and 86.91%, respectively. Sets 1 and 3 combined showed a considerable improvement with classification accuracy peaking at 96.67%



(a) Treasure Hunt

(b) Thunderbird

(c) Outlook

Figure 7: Inner-cluster variance versus k.

Table 7: Misclassification table for best feature set combination (training stage).

Application classified	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
(a) Game_setup	156432										229	
(b) Game_ctrl.		257707										
(c) Browsing	32		932493									
(d) Stor_dnld.				63212								
(e) Stor_upld.					56613							
(f) Email_mssg.						257707						
(g) Email_dir.							122343					
(h) Comms.								257552				
(i) Comms_ctrl.								87	561432		157	
(j) Streaming				35						77343		
(k) Torr_ctrl.											203764	
(l) Torrent	89											453142

with a 100-boost while even with a boost 10 or a single classifier (no boost) the prediction results were 94.52% and 92.37%, respectively.

The misclassification table generated during training stage for this best combination (sets 1 and 3) classifier is presented in Table 7. The highest number of discrepancies was observed between "game setup" and "torrent control" classes (229 flows). Due to low predictive ability (estimated during classifier training), only one attribute received packets per second (Rx.pps.) was winnowed during the training

Table 8: Flow attribute usage.

Flow attribute usage in selected C5.0 classifier		
Category	Attribute	Percentage use
Protocol and port	Protocol	80.62%
	Destination port	100%
	Source port	100%
Transmitted flow (Tx) attributes	Bytes [Tx.B.]	100%
	Packets [Tx.Pkt.]	100%
	Bits per second [Tx.bps.]	100%
	Packets per sec. [Tx.pps.]	96.25%
	Bytes per package [Tx.Bpp.]	100%
	Duration [Tx.s.]	95.48%
Received flow (Rx) attributes	Bytes [Rx.B.]	100%
	Packets [Rx.Pkt.]	100%
	Bits per sec. [Rx.bps.]	100%
	Bytes per package [Rx.Bpp.]	100%
	Duration [Rx.s.]	98.61%

stage. The remaining 14 attributes used to build the resulting classifier along with their percentage use are given in Table 8.

5.2. Confusion Matrix Analysis. The confusion matrix for selected classifier specifying cross-tabulation of predicted classes and observed values with associated statistics between different flow classes is given in Table 9. The highest errors occurred between "game control" and "browsing" flows (60114 or 1.76% of total tested flows), while no misclassification errors were observed between "game setup" and "torrent control" flows as witnessed during training cross-validation stage. The overall accuracy statistics are presented in Table 10. The value for the kappa coefficient [67, 68], which takes into account chance occurrences of accurately classified flows and is generally considered a more robust measure than simple percent agreement calculation, was also significantly high at 95.31%. The overall accuracy rate was also computed along with a 95 percent confidence interval (CI) for this rate (0.9364 and 0.956) and a one-sided test to see if the accuracy is better than the "no information rate," which is taken to be the largest class percentage in the data (P value: accuracy > NIR: $<2.2e-16$) [69]. McNemar's test P value however was not available due to sparse tables (bidirectional flow vectors having very low or zero attribute values for some flow classes, i.e., Skype control, etc.).

5.3. Sensitivity and Specificity Factor. For a given flow, the classifier's ability to accurately predict the flow class is characterized by classifier sensitivity factor and to differentiate this flow from other flow classes is by its specificity factor. Both parameters are of significant importance and ascertaining a classifier's suitability for both flow identification and discrimination. The sensitivity and specificity bar graph for each flow class for the selected classifier is given in Figure 8. Lowest

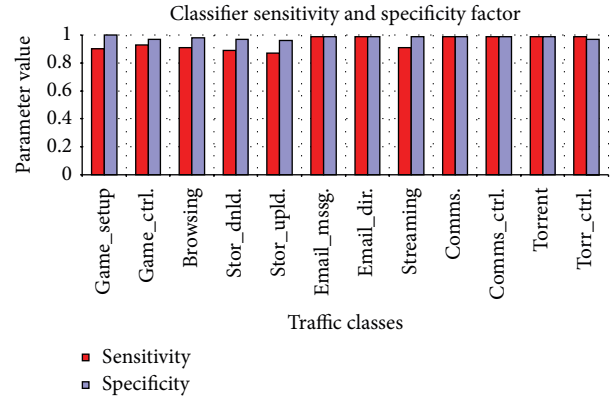


Figure 8: Classifier sensitivity and specificity factor per traffic class.

sensitivity was recorded for cloud storage flows (87.67–89.89%) among all classes, also evident from Figure 8 due to a higher mismatch between storage download and streaming (1335 or 0.039%) as well as storage upload and browsing flows (4006 or 0.11% of total tested flows). The corresponding specificity values for both storage flow classes, however, being significantly high indicated correct differentiation ability of the classifier for this application and lower sensitivity factor accredited to other application flows being misclassified under this class. Communication and BitTorrent traffic classes showed high sensitivity and specificity values. The selected classifier also showed high accuracy in detecting and differentiating between email messages and directory lookups. The classification accuracy reported per flow class was also greater than 90% for all applications apart from Dropbox which showed 87.67% accuracy due to mismatch with streaming and browsing flows. The specificity values, however, were substantively high without exception across all flow classes ranging between 98.37 and 99.57%. The results represent a highly granular classifier with ability to accurately identify application traffic as well as discriminate between flows generated by same application without employing any complex time window flow and packet analysis. As an added advantage, the approach only used a minor change in output formatting of NetFlow attributes together with basic scripting for creating bidirectional flows. The next section considers some alternate approaches for machine learning based traffic classification and compares their accuracy and computational overhead with the derived classifier.

6. Qualitative Comparison

To undertake a comprehensive qualitative evaluation of the two-phased ML approach, we considered alternate ML classifiers and appraised their viability for per-flow traffic classification in relation to the proposed technique. Weka machine learning software suite (version 3.6.13) was employed to evaluate the eight most commonly utilized supervised machine learning algorithms in comparison with the proposed two-phased approach. The comparison evaluated (i) the classification accuracy of each algorithm and (ii) the computational overhead including the training and testing times to validate

Table 9: Confusion matrix calculation for optimal classifier (evaluation stage).

Application classified	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
(a) Game_setup	156435											
(b) Game_ctrl.		257718	60114									
(c) Browsing	632	25481	932494		4006							
(d) Stor_dnld.				63208								
(e) Stor_upld.					56611							
(f) Email_mssg.						257710						
(g) Email_dir.		3981	2561				122346					
(h) Comms.								257552				
(i) Comms_ctrl.			4587						561433			
(j) Streaming				1335						77341		
(k) Torrent										2078	453143	
(l) Torr_ctrl.		5843	6154									203766

Table 10: Overall statistics.

Statistical property	Value
Classifier accuracy	96.67%
95% confidence interval (CI)	(0.9364, 0956)
No information rate	0.3332
P value (Acc > NIR)	$<2.2e-16$
Kappa	0.9531
McNemar's test P value	NA

the results from each classification technique as well as (iii) provide perspectives on the scalability of our two-phased machine learning classifier. The classifiers used the same ratio of training and testing data set pools (marked with respective application class), where 50% of the flows were used for training the respective classifier and the remaining 50% flows were used for testing purposes.

We briefly describe the machine learning algorithms that were evaluated as follows.

J48/C4.5 decision tree constructs a tree structure, in which each node represents feature tests, each branch represents a result (output) of the test, and each leaf node represents a class label, that is, application flow label in the present work [30, 70]. In order to use a decision tree for classification, a given tuple (which requires class prediction) corresponding to flow features walks through the decision tree from the root to a leaf. The label of the leaf node is the classification result. The algorithm was enabled with default parameters (confidence factor of 0.25 and reduced-error pruning by 3-fold) in the Weka implementation of the present experiment to optimize the resulting decision tree.

k nearest neighbours (kNN) algorithm computes the distance (Euclidean) from each test sample to the k nearest neighbours in the n-dimensional feature space. The classifier selects the majority label class from the k nearest neighbours and assigns it to the test sample [63]. For the present evaluation $k = 1$ was utilized.

Naïve Bayes (NB), considered as a baseline classifier in several traffic classification studies, selects optimal (probabilistic) estimation of precision values based on analysis of training data using Bayes' theorem, assuming highly independent relationship between features [60, 71].

Best-first decision tree (BFTree) uses binary splitting for nominal as well as numeric attributes and uses a top-down decision tree derivation approach such that the best split is added at each step [64]. In contrast to depth-first order in each iterative tree generation step [64, 72], the algorithm expands nodes in best-first order instead of a fixed order. Both gain and Gini index are utilized in calculating the best node in tree growth phase. The algorithm was implemented using postpruning enabled and with a default value of 5-fold in pruning to optimize the resulting classifier.

Regression tree representative (REPTree) is a fast implementation of decision tree learning which builds a decision/regression tree using information gain and variance with reduced-error pruning along with backfitting. REPTree uses regression tree logic to create multiple trees and selects the best from all the generated trees. The algorithm only sorts values for numeric attributes once. It was implemented with pruning enabled with the default value of 3-fold.

Sequential minimal optimization (SMO), a support vector classifier trained using a sequential minimal optimization algorithm by breaking optimization problem into smaller chunks, was solved analytically. The algorithm transforms nominal attributes into binaries and by default normalizes all attributes [59, 73]. It was implemented using Weka with normalization turned on along with the default parameters (the complexity parameter $C = 1$ and polynomial exponent $P = 1$).

Decision tables and naïve Bayes (DTNB) is a hybrid classifier which combines decision tables along with naïve Bayes and evaluates the benefit of dividing available features into disjoint sets to be used by each algorithm, respectively [74]. Using a forward selection search, the selected attributes are modeled using NB and decision table (conditional probability table) and at each step, and unnecessary attributes are removed from the final model. The combined model

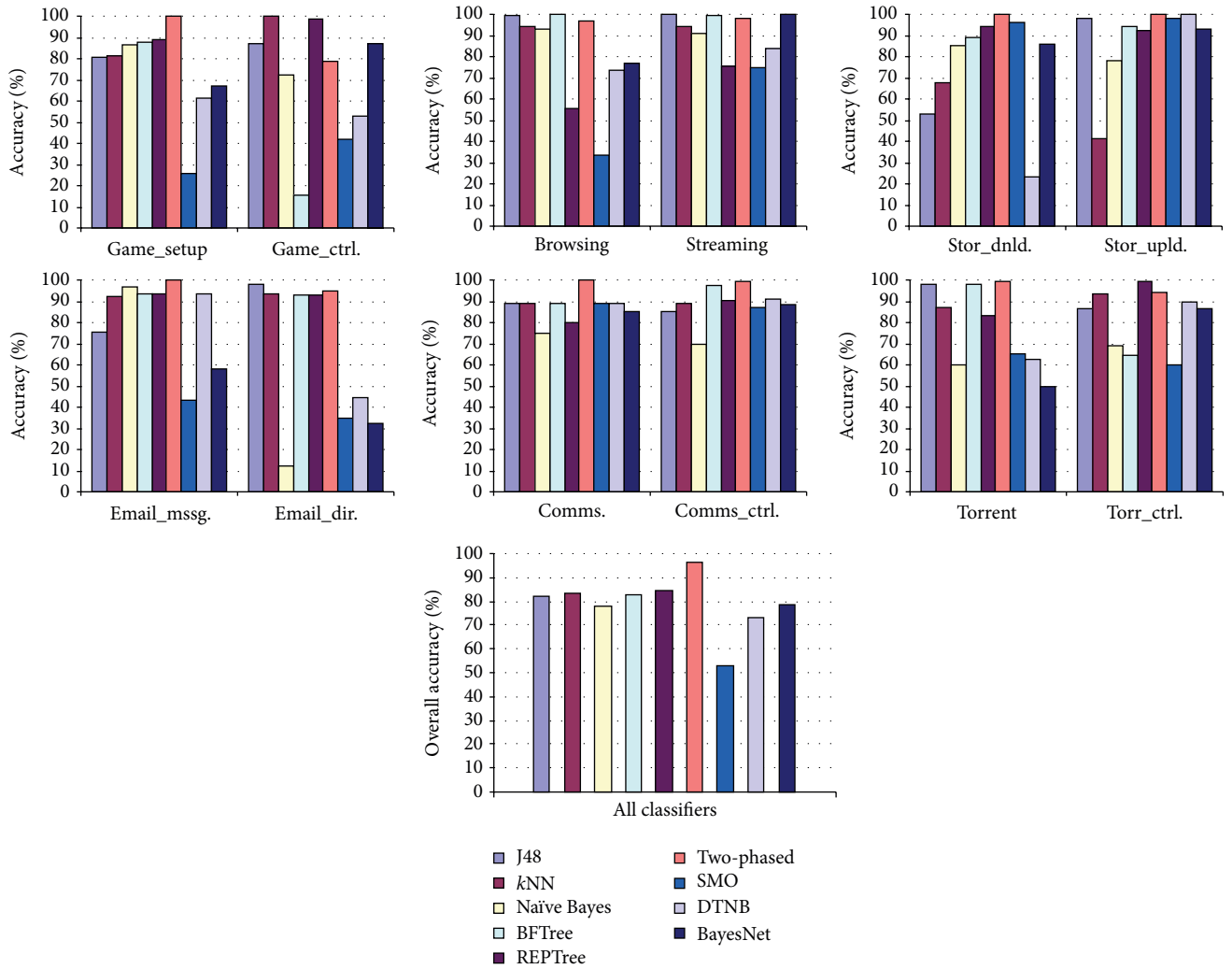


Figure 9: Comparative and average overall accuracy of machine learning algorithms for each traffic class.

reportedly [74] performs better in comparison to individual naïve Bayes and decision tables and was implemented with default parameters. The final classifier selected and used 5 attributes (out of 16 using backward elimination and forward selection search).

Bayesian network (BayesNet) is an acyclic directed graph that represents a set of features as its vertices and the probabilistic relationship among features as graph edges [62]. While using Bayes' rule for probabilistic inference, under invalid conditional independence assumption (in naïve Bayes) BayesNet may outperform NB and yield better classification accuracy [75]. The default parameters, that is, SimpleEstimator, were used for estimating the conditional probability tables of BN in the Weka implementation of BN on the training set.

The following subsections highlight a qualitative comparison between the above machine learning classification techniques and the proposed two-phased approach.

6.1. Comparative Accuracy. The respective accuracy of each examined traffic class for multiple classifiers is given in Figure 9. Overall the two-phased approach achieved better per-flow classification in comparison with the alternate techniques, while for few applications (flow types) the classification accuracy was almost equal. For the game setup flows the accuracy is the highest. For game control flows, alternate approaches such as *k*NN and REPTree provide a better percentage of correctly identified flows. This was considered earlier while evaluating the sensitivity of two-phased classifier and was mainly due to misclassification errors (of game control) with the web browsing flows. *k*NN and REPTree, however, provide a lower accuracy than two-phased ML for browsing and streaming flows. Similarly, for the streaming application tier, SMO based approach yielded highly accurate results comparable to two-phased machine learning approach while it yielded minimal accuracy when email tier was examined. For the communication application flows, almost all classifiers with the exception of NB (~63%) provided correct classification results (~80%). This was primarily due to the predictive ability of flow parameters for this set of applications. For torrent based flows, J48 decision tree along with BFTree provided almost 99.99% classification results,

with BFTree (97.25%) exceeding the two-phased classifier which gave approximately 90.02% flow identification capability of torrent control flows due to mismatch with game control and browsing flows. Therefore, while one approach might be suitable for identifying certain traffic flows, similar high accuracy might not be realized for a different application using the same classifier. In terms of overall accuracy, two-phased ML provided a much more coherent and applicable result at 96.67% with the lowest accuracy attributed to SMO at approximately 53.2% correctly classified records.

6.2. Computational Performance. To evaluate the computational performance of the classifiers, each was independently implemented on a test machine (PC), an Intel based i54310-M processor chipset with two CPUs each at 2.70 GHz and 16 GB of memory. The operating system used a GNU/Linux kernel (3.14.4 x64) and it was verified that no other user processes (apart from the Weka software suite) were consuming CPU cycles or any of the operating system processes were CPU or I/O intensive. The two-phased ML evaluation included the combined cluster analysis and subsequent C5.0 training phase from labelled flows. This was done solely to examine the computational requirements of the unsupervised and supervised machine learning ensemble, excluding the ground-truth acquisition and refinement (i.e., DPI based application flow perusal and subclass marking) which can be done offline and continuously on much greater data sets in a practical network implementation. To give a realistic comparison, the alternate classifiers used the same application labelled flows (ground-truth). The average CPU utilization for each classifier in terms of the flow records and bytes processed (testing) is given in Figure 10. We observed a linear relationship between the CPU utilization and the amount of records processed for all classifiers followed by a steady-state pattern albeit different consumption footprints. The *k*NN classifier had the highest CPU usage at up to 5.32% with a gradual decrease steadying at 4.21%. NB classifier had the lowest consumption at 1.61% while two-phased ML reported around 4.31% usage. Similarly the average memory usage per classifier in processing flow records and bytes of data is provided in Figure 11. The BFTree algorithm had the highest memory usage at 190.28 MB with the two-phased ML at 175.31 MB. BayesNet had the lowest memory footprint with a steady-state value of approximately 50.14 MB.

The average training and testing times with respect to three different sizes of flow sets (1000, 1 million, and 3 million) for each classifier are depicted in Figure 12. The training time for two-phased classifier was significantly high compared to other classifiers for flow record size of 1000 flows. This was due to the in-tandem processing of the two embedded algorithms used. The training time relationship for most classifiers with respect to the size of training data at larger values of the latter was, however, nonlinear. The training time for J48, for example, for both 1M and 3M flows, was approximately the same averaging at around 59.35 minutes. Similarly, BFTree approximated in between 60.12 minutes for 1M and 63.45 minutes for 3M flows, respectively. Two-phased classifier also reported between 80.87 minutes and 84.51 minutes for the respective flow records in the training phase. This yields

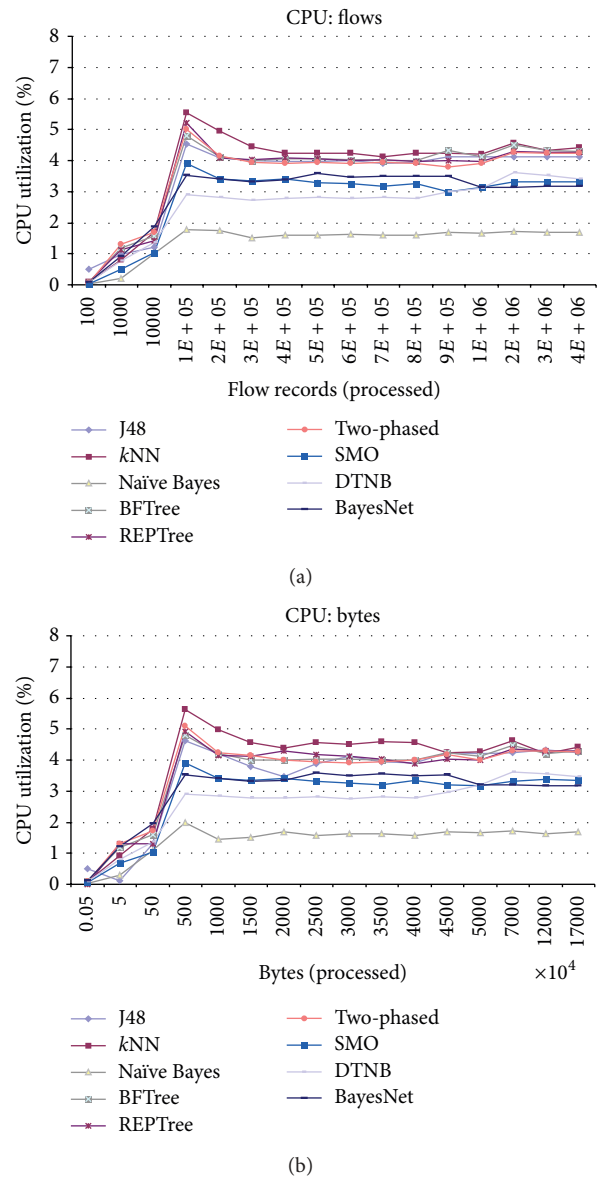


Figure 10: CPU utilization percentage: (a) flow records; (b) bytes.

approximately on average 0.88 seconds spent training around 1K flows with a standard deviation (σ) of 1.137 between 1M and 3M flows. Hence, the proposed technique results in better performance in terms of training times in the steady state with relatively larger data sets. However, as noted above it does not specifically consider the time duration involved in offline analysis of optimal cluster labelling following examination of different types of traffic generated per application. The SMO classifier accounted for the highest training times with larger flow records requiring around 140.35 minutes of training 3M flows. SMO, therefore, reported the lowest accuracy while having substantial resource consumption, performing quite marginally compared to other techniques.

Considering the testing timelines, NB followed by J48 classifiers were most efficient in classifying flows at approximately 6.3 minutes and 8.12 minutes, respectively.

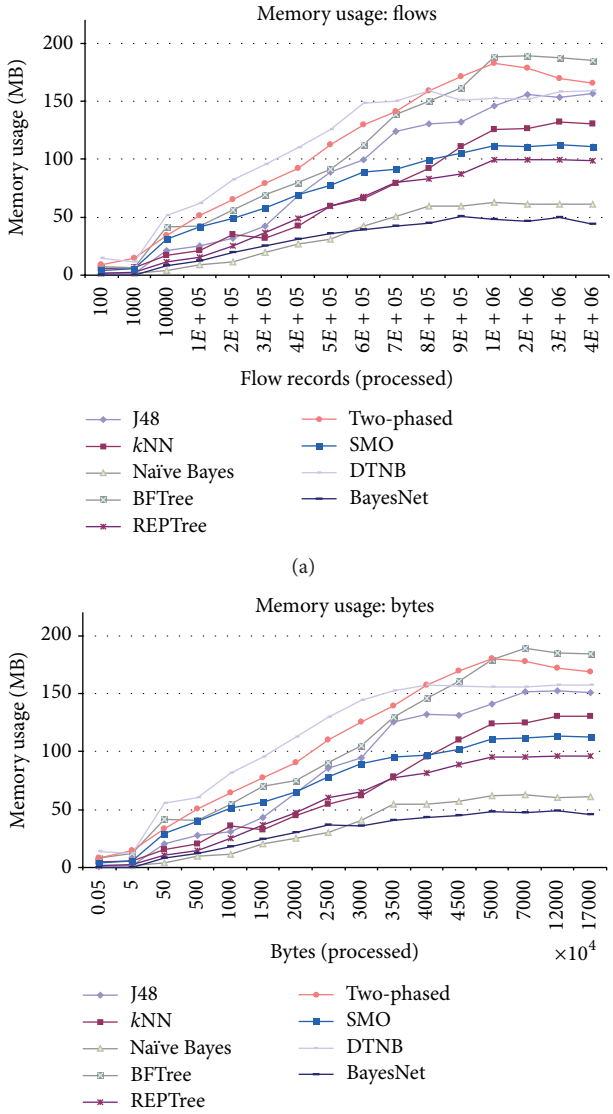


FIGURE 11: Memory usage: (a) flow records; (b) bytes.

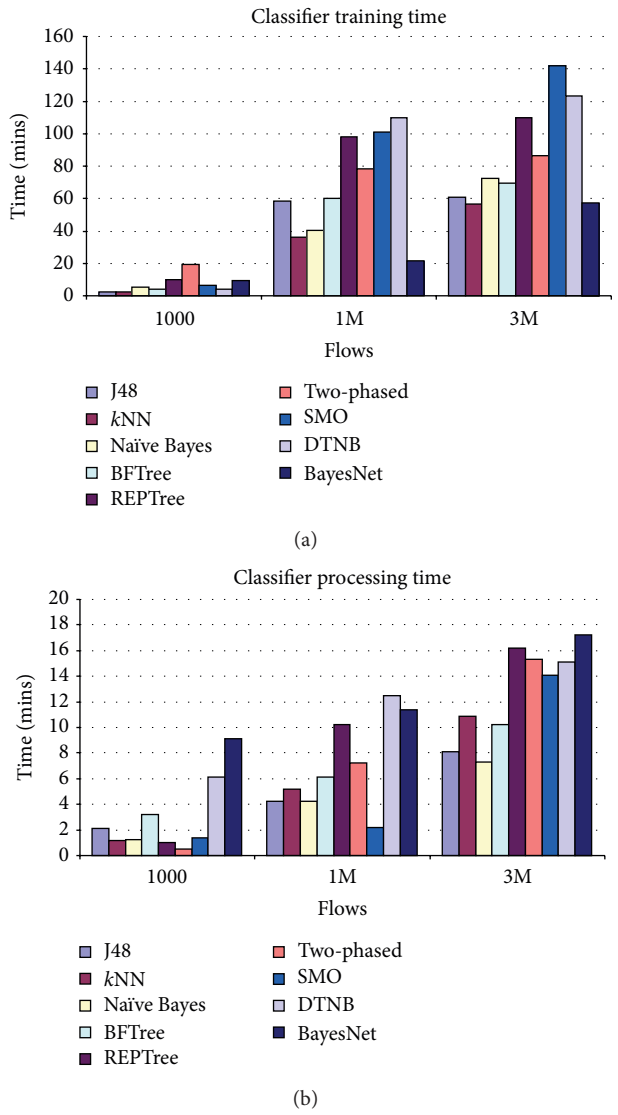


FIGURE 12: Classifier time frames for (a) training and (b) processing time.

Two-phased recorded a linear relationship between the flows tested and the respective processing time frame. Approximately 15.17 minutes was spent in classifying 3M flows, averaging at 0.30 seconds for processing 1K flow records with a standard deviation (σ) of 0.071 between 1M and 3M flows. Thus, given the high accuracy of the two-phased approach the computation performance seems highly applicable in realistic traffic classification scenarios. BN reported the highest 16.91 minutes in testing 3M flows albeit average overall classification performance as depicted in Figure 9. The two-phased approach therefore yields better accuracy across all traffic classes with a comparably smaller computational cost when considered in relation to the examined alternate classification approaches implemented using the Weka platform. However, it may be noted that since Weka is a Java based implementation of the classifiers, the exact computational overhead reported might be different when a standalone classifier utility for each approach is applied resulting in a more efficient performance.

6.3. Scalability. In classification accuracy comparisons among several classifiers it is evident that the prediction ability of a scheme is highly dependent on analysing a correct measure of variation between the selected flow attributes for each traffic class. Traditionally the bidirectional flow features utilized in the present research have shown considerable applicability in multiple classifiers to attain a (somewhat) acceptable degree of traffic identification. However, as highlighted in [14, 16] the wide majority of the classification algorithms are infeasible with respect to their application in the network backbone by ISPs. The reasons for this lack of applicability range from the tremendous amount of traffic generated in the network core to the actual methodology of

the approach, for example, sometimes requiring analysis of end-point behaviour for classification [16, 76]. In addition flow based techniques often rely on statistical information from bidirectional traffic (specifically TCP) and placing the traffic measurement or collection point as close to the ingress or the edge of the network as possible to collect the necessary features from outbound as well as inbound flows. An alternate approach to address this limitation [77] details an algorithm for predicting the inbound traffic flow attributes based on the unidirectional transmitted TCP flows. However, in the present case we propose using the former technique of keeping flow measurements as close to the ingress or edge of the network. This technique ensures corroboration between upstream and downstream host traffic to generate bidirectional flow features, minimizing the operational and computational cost of implementing the two-phased classifier.

The proposed two-phased approach is significantly reproducible due to the utilization of NetFlow, ubiquitous in present ISP networking gear. Additionally, the derived classifier reported high efficiency in dealing with voluminous data (flow records) with high level of accuracy, again a basic traffic classification requirement by service providers. The synergetic combination of classifiers in the present case produced comprehensive traffic classification results and a comparatively lower processing overhead while using non-specialized hardware.

7. Conclusion

The present paper used a twofold machine learning approach for traffic classification on a per-flow basis by solely using NetFlow attributes and without depending on packet derivatives or complex time window analysis. During the unsupervised phase, approximately 6.8 million bidirectional flows for all applications were collected and cluster analysed resulting in 12 unique flow classes. The supervised phase used four different feature sets of NetFlow attributes from the derived flow classes to test and train the C5.0 ML decision tree classifier. The foremost feature set comprising 14 NetFlow attributes reported an average prediction accuracy of 92.37% increasing to 96.67% with adaptive boosting. Sensitivity factor of the classifier was also exceedingly high ranging above 90% with only cloud storage flows (file upload and downloads) reporting relatively low values between 87.67 and 89.89% due to misclassification with general web browsing and streaming flows. The corresponding specificity factor, however, translating for classifier flow discrimination ability ranged between 98.37 and 99.57% across all applications. Furthermore, the substantive accuracy of the present approach in achieving highly granular per-flow application identification and the computational efficiency in comparison with other machine learning classification methodologies paves way for future work in extending this method to include other applications for real-time or near real-time flow based classification.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

References

[1] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Computer Networks*, vol. 76, pp. 75–89, 2015.

[2] R. Sadre, A. Sperotto, R. Hofstede, and N. Brownlee, "Flow-based approaches in network management: recent advances and future trends," *International Journal of Network Management*, vol. 24, no. 4, pp. 219–220, 2014.

[3] M. Iliofotou, B. Gallagher, T. Eliassi-Rad, G. Xie, and M. Faloutsos, "Profiling-by-association: a resilient traffic profiling solution for the Internet backbone," in *Proceedings of the 6th International Conference on Emerging Networking Experiments and Technologies (Co-NEXT '10)*, Philadelphia, Pa, USA, December 2010.

[4] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[5] J. Camacho, P. Padilla, P. García-Teodoro, and J. Díaz-Verdejo, "A generalizable dynamic flow pairing method for traffic classification," *Computer Networks*, vol. 57, no. 14, pp. 2718–2732, 2013.

[6] A. Dainotti, A. Pescapè, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.

[7] L. Stewart, G. Armitage, P. Branch, and S. Zander, "An architecture for automated network control of QoS over consumer broadband links," in *Proceedings of the IEEE Region 10 International Conference (TENCON '10)*, November 2005.

[8] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "KISS: stochastic packet inspection classifier for UDP traffic," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1505–1515, 2010.

[9] P. Bermolen, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Abacus: accurate behavioral classification of P2P-TV traffic," *Computer Networks*, vol. 55, no. 6, pp. 1394–1411, 2011.

[10] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2nd Conference on Future Networking Technologies (CoNEXT '06)*, Lisboa, Portugal, December 2006.

[11] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.

[12] A. Dainotti, A. Pescapé, and C. Sansone, "Early classification of network traffic through multi-classification," in *Traffic Monitoring and Analysis*, J. Domingo-Pascual, Y. Shavitt, and S. Uhlig, Eds., vol. 6613 of *Lecture Notes in Computer Science*, pp. 122–135, Springer, Heidelberg, Germany, 2011.

[13] T. Z. J. Fu, Y. Hu, X. Shi, D. M. Chiu, and J. C. S. Lui, "PBS: periodic behavioral spectrum of P2P applications," in *Passive and Active Network Measurement*, S. B. Moon, R. Teixeira, and S. Uhlig, Eds., vol. 5448 of *Lecture Notes in Computer Science*, pp. 155–164, Springer, Berlin, Germany, 2009.

[14] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the ACM CoNEXT Conference (CoNEXT '08)*, 12 pages, New York, NY, USA, 2008.

[15] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability

of classification schema," *Computer Networks*, vol. 53, no. 6, pp. 790–809, 2009.

[16] S. Valenti, D. Rossi, A. Dainotti, A. Pescape, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*, vol. 7754 of *Lecture Notes in Computer Science*, pp. 123–147, Springer, 2013.

[17] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.

[18] S. Kumar and P. Crowley, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '06)*, pp. 339–350, Pisa, Italy, September 2006.

[19] T. Karagiannis, A. Broido, M. Faloutsos, and K. C. Claffy, "Transport layer identification of P2P traffic," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC '04)*, pp. 121–134, Taormina, Italy, October 2004.

[20] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos, "Profiling the end host," in *Passive and Active Network Measurement*, S. Uhlig, K. Papagiannaki, and O. Bonaventure, Eds., vol. 4427 of *Lecture Notes in Computer Science*, pp. 186–196, Springer, Heidelberg, Germany, 2007.

[21] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, vol. 35, no. 4, pp. 169–180, ACM, 2005.

[22] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs (TDGs)," in *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference (IMC '07)*, pp. 315–320, San Diego, Calif, USA, October 2007.

[23] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–34, 2012.

[24] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Tech. Rep., University of Cambridge, Cambridge, UK, 2005.

[25] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the ACM SIGCOMM Workshop on Mining Network Data (MineNet '06)*, pp. 281–286, ACM, New York, NY, USA, September 2006.

[26] L. Yingqiu, L. Wei, and L. Yunchun, "Network traffic classification using K-means clustering," in *Proceedings of the 2nd International Multi-Symposiums on Computer and Computational Sciences (IMSCCS '07)*, pp. 360–365, August 2007.

[27] V. Carela-Español, P. Barlet-Ros, and J. Solé-Pareta, "Traffic classification with sampled NetFlow," Tech. Rep. UPC-DAC-RR-CBA-2009-6, 2009.

[28] D. Rossi and S. Valenti, "Fine-grained traffic classification with Netflow data," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10)*, pp. 479–483, July 2010.

[29] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, "Internet traffic classification using constrained clustering," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2932–2943, 2014.

[30] A. B. Mohammed and S. M. Nor, "Near real time online flow-based internet traffic classification using machine learning (C4.5)," *International Journal of Engineering*, vol. 3, no. 4, pp. 370–379, 2009.

[31] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 machine learning algorithm," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '12)*, pp. 237–241, Maui, Hawaii, USA, February 2012.

[32] O. Mula-Valls, *A practical retraining mechanism for network traffic classification in operational environments [M.S. thesis in Computer Architecture, Networks and Systems]*, Universitat Politècnica de Catalunya, Barcelona, Spain, 2011.

[33] P. Foremski, C. Callegari, and M. Pagano, "Waterfall: rapid identification of ip flows using cascade classification," *Communications in Computer and Information Science*, vol. 431, pp. 14–23, 2014.

[34] V. Carela-Español, P. Barlet-Ros, M. Sole-Simo, A. Dainotti, W. de Donato, and A. Pescape, "K-dimensional trees for continuous traffic classification," in *Traffic Monitoring and Analysis: Second International Workshop, TMA 2010, Zurich, Switzerland, April 7, 2010.Proceedings*, vol. 6003 of *Lecture Notes in Computer Science*, pp. 141–154, Springer, Berlin, Germany, 2010.

[35] W. de Donato, A. Pescape, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014.

[36] S. Lee, H. Kim, D. Barman et al., "NeTraMark: a network traffic classification benchmark," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 22–30, 2011.

[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[38] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*, MIT Press, Cambridge, Mass, USA, 2006.

[39] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semisupervised network traffic classification," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '07)*, San Diego, Calif, USA, June 2007, *Performance Evaluation Review*, vol. 35, no. 1, pp. 369–370, 2007.

[40] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2015.

[41] L. Salgarelli, F. Gringoli, and T. Karagiannis, "Comparing traffic classifiers," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 65–68, 2007.

[42] F. Bakerand, B. Fosterand, and C. Sharp, "Cisco architecture for lawful intercept in IP networks," RFC 3924, IETF, 2004.

[43] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, Berkeley, Calif, USA, 1967.

[44] B. S. Everitt and T. Hothorn, *A Handbook of Statistical Analyses Using*, Chapman & Hall/CRC, Boca Raton, Fla, USA, 2006.

[45] Ntopng Traffic Classifier, http://www.ntop.org/products/traffic-analysis/ntop/.

[46] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '05)*, pp. 229–240, Philadelphia, Pa, USA, August 2005.

[47] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: automated construction of application signatures," in *Proceeding of the ACM SIGCOMM Workshop on Mining Network Data (MineNet '05)*, pp. 197–202, ACM Press, August 2005.

[48] Libcaplibrary, http://www.tcpdump.org/.

[49] SoftflowDaemon, http://www.mindrot.org/projects/softflowd/.

[50] Nfdump Tool Suite, http://nfdump.sourceforge.net/.

[51] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network—measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[52] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: a youtube-based measurement study," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[53] Skype, "What are P2P Communications?" https://support.skype.com/en/faq/FA10983/what-are-p2p-communications.

[54] S. A. Baset and H. G. Schulzrinne, "An analysis of the Skype peer-to-peer internet telephony protocol," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–11, Barcelona, Spain, April 2006.

[55] PRTG Network Monitor, https://www.paessler.com/prtg.

[56] Nagios IT Infrastructure Monitoring, https://www.nagios.org/.

[57] G. Bossert, F. Guihéry, and G. Hiet, "Towards automated protocol reverse engineering using semantic information," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '14)*, pp. 51–62, 2014.

[58] J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: automatic extraction of protocol message format using dynamic binary analysis," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 317–329, Alexandria, Va, USA, October 2007.

[59] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Advances in Kernel Methods—Support Vector Learning, 1998.

[60] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd International Conference on Machine learning (ICML '06)*, pp. 161–168, ACM, 2006.

[61] W. Li, K. Abdin, R. Dann, and A. Moore, *Approaching Real-Time Network Traffic Classification (ANTCs)*, RR-06-12, Department of Computer Science Research Reports, Queen Mary College, University of London, 2006.

[62] J. Pearl, "Bayesian networks: a model of self-activated memory for evidential reasoning," in *Proceedings of the 7th Conference of the Cognitive Science Society*, UCLA Technical Report CSD-850017, pp. 329–334, University of California, Irvine, Calif, USA, August 1985.

[63] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[64] H. Shi, *Best-first decision tree learning [M.S. thesis]*, University of Waikato, Hamilton, New Zealand, 2007.

[65] T. Sinam, I. T. Singh, P. Lamabam, and N. Ngasham, "An efficient technique for detecting Skype flows in UDP media streams," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS '13)*, pp. 1–6, IEEE, Kattankulathur, India, December 2013.

[66] P. Casas and R. Schatz, "Quality of experience in cloud services: survey and measurements," *Computer Networks*, vol. 68, pp. 149–165, 2014.

[67] J. Cohen, "Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit," *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.

[68] J. Carletta, "Assessing agreement on classification tasks: the kappa statistic," *Computational Linguistics*, vol. 22, no. 2, pp. 248–254, 1996.

[69] M. Kuhn, "Building predictive models in R using the caret package," *Journal of Statistical Software*, vol. 28, no. 5, 2008.

[70] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[71] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pp. 338–345, Morgan Kaufmann, 1995.

[72] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[73] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.

[74] M. Hall and E. Frank, "Combining naive bayes and decision tables," in *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, D. L. Wilson and H. Chad, Eds., pp. 318–319, AAAI Press, Miami, Fla, USA, May 2008.

[75] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, New York, NY, USA, 1989.

[76] G. Szabo, I. Szabo, and D. Orinscay, "Accurate traffic classification," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '07)*, pp. 1–8, IEEE, Espoo, Finland, June 2007.

[77] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 883–892, Banff, Canada, May 2007.