Master of Science (Computer Science)

Analysis and Application of Machine Learning Techniques for Statistical Network Flow Classification

THE UNIVERSITY OF MELBOURNE

Department of Computing and Information Systems,

The University of Melbourne, Australia

Author: Timothy Andrew Glennan

Student number:

Primary Supervisor: Christopher Andrew Leckie

Co-Supervisor: Sarah Monazam Erfani

October 2016

Declaration

I certify that

- this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text;
- clearance for this research from the University's Ethics Committee was not required; and
- the thesis is 22,807 words in length (excluding text in images, tables, bibliographies and appendices).

Abstract

Modern network traffic classification approaches apply machine learning techniques to statistical flow properties, allowing accurate classification even when traditional approaches fail. We first consider the use of an important alternative flow format that is not well evaluated in the literature. Through a combination of supervised learning and visualization, we identify core weaknesses in the flow format that make it unsuitable for statistical traffic classification. We also propose a novel approach to semi-supervised traffic classification to classify both known and unknown flows with minimal pre-labelled training data. In our model, we propose a new algorithm for mapping clusters to classes in order to target previously difficult-to-classify classes. Furthermore, we introduce a conservative methodology for applying feature selection when unknown traffic classes are present, and we propose a compound classification approach with sampling to significantly improve classification efficiency. We find our approach can achieve overall accuracy upwards of 94.17%, over 16% above the state-of-the-art technique on which it is based. Additionally, our approach generally improves the classification performance on any traffic class. Finally, we introduce an original system for reducing the complexity of unsupervised clustering approaches. This system combines a novel equivalence metric for grouping flows and proposes a cluster similarity function for cluster aggregation. Our approach is proposed in response to the cluster-application identification problem, which currently represents the key weakness of existing unsupervised approaches.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Christopher Leckie and Sarah Erfani, both of whom went above and beyond expectations throughout the entirety of my research project. Their ongoing encouragement, support, and guidance were paramount to both the work presented in this thesis and to the expansion of my personal knowledge.

Contents

List of Tables

List of Figures

Chapter 1
Introduction

Classifying network traffic has important applications across a wide range of common networking tasks, including network management, surveillance, and security. Accurately classifying network applications is an important challenge, which is complicated by a continuously increasing number of applications appearing on networks [1], many of these often defying ratified standards and even attempting to mask their presence [2].

Traditionally, network traffic classification has been achieved by inspecting port numbers. While this method is both simplistic and historically sufficient, it is no longer considered a reliable method of traffic classification [3]. A growing number of applications use non-standard port numbers, causing port number classification to be ineffective [3]-[4]. For example, applications may use non-standard port numbers as a simple approach for masking activity [4]-[5]. Peer-to-peer (P2P) traffic is one such example of an application that commonly utilises non-standard ports for this reason [5]. Other applications use commonly open ports in attempts at more reliable connections. Since firewalls often block certain port ranges, these applications may simply share the same port number with existing applications, or even use the existing application as a tunnelling transport layer [4]. Furthermore, many applications do not have standard port numbers, may use the same port numbers as other applications, and may even dynamically allocate ports [4]. These prevalent issues all contribute to the unreliability of using port numbers for traffic classification.

Deep-packet inspection (DPI) is a popular and effective alternative to the traffic classification problem [3]. DPI compares signatures of packet payloads against a database of known application signatures to effectively classify traffic. While avoiding a reliance on port numbers, DPI is instead heavily reliant on database of application signatures. This database must therefore be well maintained for reliable classification. Identifying applications with DPI also incurs significant computational complexity, which can make DPI unsuitable in some use cases [2], [6]. Additionally, packet payloads cannot always be inspected. For example, payloads may not be captured for privacy reasons or due to the sheer volume of network data. Furthermore, DPI will fail when traffic is encrypted [7].

Motivated by the drawbacks of these traditional techniques, machine learning approaches have been gaining popularity for their ability to classify network applications using exclusively statistical features [1], [6]-[8]. Rather than classify individual packets, machine learning approaches instead classify a series of packets representing a connection between two hosts. This connection is called a traffic flow.

We define a "flow" here as a connection between two hosts sharing the same 5-tuple: source IP, source port, destination IP, destination port, and protocol. From such flows, statistical properties such as the mean packet size and mean time between packets can be utilised to model applications purely by usage patterns. The use of purely statistical features allows the identification of traffic applications even when non-standard port numbers are used, when traffic is encrypted, and when the payload is unavailable. Additionally, these approaches avoid reliance on maintaining databases such as is the case with in DPI. Aside from preserving privacy for users [9], using statistical flow features instead of packets benefits machine learning approaches in two main ways. Firstly, both packets and flows are considered big data [10]. That is, the high volume and velocity with which network data is continually generated causes inherent difficulty in its analysis. However, flows are aggregations of packets and thus allow learning on a wider variety of data. Secondly, the higher-level view provided by flows allows more detailed statistical perspectives on the ways network applications interact with hosts. Machine learning approaches can therefore build sufficiently strong statistical feature sets without requiring unreliable features such as port numbers. Various unsupervised [11]-[16], supervised [17]-[22], and semi-supervised [6], [8], [23]-[24] machine learning approaches have demonstrated the suitability of using statistical features for the network traffic classification problem.

The work in this thesis addresses the open problem of how to improve existing traffic classification methods using only statistical traffic flow properties. We apply supervised, unsupervised, and semi-supervised techniques, considering both existing strengths and ways to improve each set of techniques for the problem. The remainder of the thesis is structured as follows. Chapter 2 presents a review of related machine learning approaches for network traffic classification. Chapter 3 describes the datasets used throughout these works. Throughout Chapter 4, an exploratory analysis is conducted to understand the viability of machine learning approaches for a flow format that is not well explored in the literature. This analysis combines supervised learning and visualisation techniques to explore the relative strengths of multiple flow formats. In Chapter 5, we consider existing semi-supervised clustering approaches for classifying network flows. We focus on these approaches for their realistic requirements of minimal pre-labelled training data. We identify a core weakness in these approaches, in response to which we propose a number of innovations for training more effective and stable flow classifiers. In Chapter 6, we consider the main drawback of existing unsupervised approaches for traffic classification. We propose a novel system that exploits domain-specific properties to reduce model complexity and therefore ease the interpretation of unsupervised approaches. Finally, our conclusions and future directions are presented in Chapter 7.

The main contributions of this work are as follows:

- Our exploratory analysis demonstrates that not all flow formats have sufficient resolution to be classified with statistical properties. We exhibit the clear strength and value of a rich statistical feature set, which affords clear separation of traffic classes and generalizability. The lack of variation found in alternative formats motivates the capture of more complex flows.
- We propose a number of innovations to semi-supervised flow classifiers, including an original algorithm for mapping clusters to traffic classes. Based on an empirical evaluation on a standard benchmark dataset, we demonstrate that our semi-supervised flow classification approach can achieve upwards of 94% accuracy, an increase of more than 16% against two leading techniques. We demonstrate how our approach is more stable, and that it can improve the classification performance for all traffic applications.
- We demonstrate how sampling can be used to reduce classification time by over 80% with minimal impact on effectiveness, which has important implications for timely decision-making.
- Our novel cluster aggregation approach exhibits strong potential for countering the main weakness of unsupervised approaches: poor model interpretability. We demonstrate how the effort required to interpret existing unsupervised systems can be significantly reduced. We also reason how our system can be used as a pre-processing step for realistic supervised learning approaches.

The following paper was published based on the work presented in Chapter 5:

- T. Glennan, C. Leckie, and S. M. Erfani, "Improved classification of known and unknown network traffic flows using semi-supervised machine learning," in *The 21st Australasian Conference on Information Security and Privacy (ACISP)*, 2016, pp. 493-501.

Chapter 2
Literature Review

An extensive range of research has demonstrated the effectiveness of combining statistical flow features and machine learning algorithms to classify network traffic. Supervised, unsupervised, and semi-supervised techniques have all been shown to be viable, each set of techniques having their own strengths and weaknesses. In this chapter, an overview of foundational and state-of-the-art approaches for each set of techniques is given. Additionally, we present a brief survey of approaches that use alternative flow feature sets.

2.1 Unsupervised Approaches

An early framework for unsupervised traffic classification was proposed in [11]. With a focus on identifying a strong set of statistical features to discriminate flows, the unsupervised setting was motivated by the ability to cluster without a-priori knowledge of the traffic classes. The framework involved the derivation of statistical flow features through the conversion of packet-level network traffic. These statistics could then naturally be clustered using AutoClass – a variant of the classic Expectation-Maximization (E-M) algorithm – and used as a nearest cluster classifier. The proposed framework was evaluated in [12], where the clustering algorithm was applied to the strongest statistical feature set obtained through feature selection techniques. Reporting an average of 86.5% accuracy across multiple datasets, the key contribution of this early work was the demonstrated potential of clustering flow statistics. Furthermore, feature selection techniques revealed that statistical features involving packet sizes were typically good discriminators between traffic classes. Temporal features were concluded to be less important. While this approach is simplistic and an effective foundation to unsupervised traffic classification approaches, it was found that clustering on statistical flow features did not guarantee a single cluster per traffic class. Instead, it was found that some classes such as FTP and HTTP traffic had very diverse statistical characteristics and thus were split into many smaller clusters. Meanwhile, the Telnet class heavily overlapped with all other classes. The disparities observed between the clustering and true traffic classes are found to be the main of the unsupervised system.

In [13], Erman, Arlitt, and Mahanti evaluated three classic clustering algorithms for unsupervised flow classification. As in [12], the AutoClass algorithm was used. Additionally, this work is the first to consider applying the *k*-Means and DBSCAN algorithms to the problem. While all three algorithms were reported to effectively cluster statistical flow features, the AutoClass implementation was found to have the highest effectiveness. Across two datasets, an average of 90.55% accuracy was reported for

AutoClass. It was further highlighted that this algorithm has a very notable advantage of automatically determining the number of clusters. However, AutoClass was also shown to be significantly slower than the alternatives. On the same data, the AutoClass algorithm required 4.5 hours to complete, compared to just 1 minute and 3 minutes for *k*-Means and DBSCAN respectively. With an average of 73.8% accuracy, DBSCAN was determined to be least suitable for flow data. Despite the reduced accuracy, it was found that DBSCAN produces the most accurate clusters due to the noise-handling capabilities of the algorithm. DBSCAN automatically separates noise from clusters, whereas the *k*-Means and AutoClass clusters include all data points. The low accuracy was therefore attributed to noisy data points being regarded as misclassifications. However, DBSCAN clusters typically had higher precision for each traffic class, demonstrating its highly accurate clustering. With an appropriate number of clusters, *k*-Means reported mid-range effectiveness at 81.50% accuracy. It was determined that a reasonable number of clusters tended to be between 100 and 150 for just 8 traffic applications, and that using up to 500 clusters could result in further improvements. However, selecting an appropriate number of clusters was concluded to be the main drawback of *k*-Means. With too few clusters, the accuracy was shown to be very poor. However, the significantly reduced training time is much more appropriate for the large volumes of data in networking contexts. In the comparisons conducted in this work, a total of just 8,000 network flows were used to accommodate the slow train time of AutoClass. The *k*-Means algorithm can handle significantly more data, and is thus more realistic in the network context. For both its classification effectiveness and fast training time, *k*-Means was concluded to be the most appropriate clustering algorithm for statistical flow classification.

A number of similar unsupervised clustering approaches have further demonstrated the strength of clustering on statistical flow features. In [14], the AutoClass algorithm is shown to be able to outperform the supervised Naïve Bayes algorithm. With 91% accuracy, the AutoClass clustering led by a significant 9%. This work had two main contributions. Firstly, the difference in accuracy demonstrated the value of supporting previously unknown applications through clustering. The inability for the supervised algorithm to handle unknown applications was a major factor in its lower performance. Secondly, the authors identified that each cluster typically represented a single traffic application. Thus, analysis could be simplified by manually analysing only a sample of data points in each cluster to identify applications. In [15], Wang et al. utilise properties unique to Internet traffic to constrain their unsupervised clusters. In this model, the authors proposed clustering such that flows cannot be clustered together without satisfying equivalence constraints specific to this domain. These constraints combined knowledge of Internet protocols and timestamps. The constraints resulted in up to 94% and 97% accuracy across their two datasets, improvements of 4% and 11% over E-M and *k*-

Means. Not only did this outperform the *k*-Means and E-M algorithms, but the constrained clustering algorithm was also found to converge more quickly than *k*-Means. The constrained clustering algorithm also tended to require fewer clusters than *k*-Means and E-M, but this number was still significantly larger than the true number of classes. Therefore, in these and other unsupervised approaches, mapping clusters to specific applications remains a significant challenge due to the consistently demonstrated requirement of a large number of clusters for purity. In these approaches, cluster labelling requires manual analysis or supplementary information.

In attempts to reduce the disparity between the number of clusters and classes, the system proposed in [16] automatically aggregates clusters belonging to the same traffic application. The proposed approach combines *k*-Means clustering with payload analysis to merge clusters with similar packet payloads. Using a bag-of-words model to represent the payloads of flows in each cluster, similar clusters can be identified through their payloads. Latent semantic analysis then hierarchically merges clusters. The authors propose to merge clusters until either a predefined number of clusters is reached, or no two clusters exceed some similarity threshold. However, a similarity threshold was not explored here. By predefining a cluster limit equal to the true number of classes, the authors showed the ability to aggregate clusters to achieve 89% classification accuracy. This accuracy was just 2% below the accuracy of the clustering prior to aggregation. The approach was also shown to outperform six supervised classifiers (C4.5 decision trees, k-Nearest Neighbours, Support Vector Machines, Naïve Bayes, Neural Networks, and Bayes Nets), with approximately 5% higher accuracy than the strongest of these approaches (Bayes Nets). However, the authors used just 1,000 labelled flows to train each supervised classifier, and all approaches were then evaluated on 20,000 held-out testing flows. For the 13 applications included here, 1,000 training flows is a small and likely insufficient pool of training data. Since previous approaches have demonstrated that some applications have very diverse characteristics [11], we do not expect that this sufficiently captured the expected range of characteristics in the testing flows. Thus the relative effectiveness reported likely underrepresents the true comparative strength of supervised models. The proposed approach clusters 10,000 unlabelled flows, which is more likely to capture patterns in the testing flows that were unseen by the supervised systems. Additionally, the requirement for the true number of classes for aggregation requires external knowledge that can be considered impractical for real-world implementations, especially due to unknown classes. Finally, while the authors demonstrate value from utilising packet payloads, the effects that encrypted traffic would have on this method are not explored.

The extremely large volumes of traffic generated by networks combined with the difficulty of obtaining sufficient labelled data [15] make unsupervised approaches a natural fit for traffic classification. The works outlined in this section demonstrated the strength of these approaches. Additionally, the inherent ability to identify new and previously unknown applications by clustering novel patterns [14] is a natural advantage over supervised approaches. Despite the range of works reporting high effectiveness, the large disparity between the number of clusters and traffic classes remains a key challenge for unsupervised approaches.

In the following section, we survey supervised approaches for the same problem.

2.2 Supervised Approaches

Supervised machine learning approaches use fully labelled training samples to build classification models, which are then used to predict the class of new samples. These are generally expected to outperform unsupervised and semi-supervised approaches when sufficient labelled data is available [17]. Five classic supervised learning algorithms are applied to traffic classification in [18], where each algorithm is evaluated in terms of both effectiveness and efficiency. C4.5 decision trees, Bayesian Networks, Naïve Bayes Tree, and two variants of Naïve Bayes (discretisation and kernel density estimation) are compared for their relative suitability for the problem. Four of the algorithms achieved above 95% accuracy, with C4.5 decision trees obtaining the highest accuracy by a small margin. The kernel density estimation variant of Naïve Bayes performed relatively poorly in terms of effectiveness, with approximately 80% accuracy. The authors attributed the significant decrease in accuracy to the specific traffic classes used, as the same algorithm was previously shown to perform well in other works. While the best four algorithms were approximately equivalent in terms of overall effectiveness, there was variation in training and classification efficiency. The Naïve Bayes Tree was an outlier in terms of training time, requiring the longest time by a large margin. The remaining algorithms were more equal in terms of training time. For classifying new flows, the C4.5 decision trees were clearly the fastest, followed by the discretisation Naïve Bayes and Bayesian Networks. Notably it was also found that reducing the feature set improved the classification speed on all classifiers except the decision trees. In these experiments, it was found that a smaller feature set could actually increase the classification time for the C4.5 decision trees by a small margin due to the construction of larger trees. Despite this, C4.5 still clearly outpaced the remaining algorithms. The main finding of this work was therefore that, out of the algorithms considered here, C4.5 decision trees are the most appropriate supervised learner for traffic classification. While even standard implementations of each algorithm

was shown to be an effective flow classifier, the decision trees achieved the highest accuracy with the best classification time, and did so with good training time.

With a focus on improving the efficiency of network management, Nguyen et al. proposed a novel use of traffic flows in [19]. For efficient network management, the authors propose classifying "sub-flows", subsections of flows of up *N* packets long. The key advantage the approach that an effective sub-flow classifier could classify a new flow at any point in its lifetime. Most significantly this includes the classification of flows in real-time, even before flows are complete. The sub-flow system begins by segmenting regular flows into many *N*-length sub-flows using a sliding window approach. Supervised classifiers are then trained on these. However, sub-flows introduce a directionality issue. Flows are commonly seen as bidirectional, with the forward direction being the host from which the first packet is sent. As sub-flows can begin at any point in a connection's lifetime, directionality is not reliably preserved during classification. To account for this, the authors propose training on complimentary pairs of flows. That is, an equivalent flow in the reverse-direction is created for each flow in the training dataset. Training on sub-flows in each direction thus resolves the directionality issue. However, sub-flows significantly increase the number of training samples compared to regular flows, and the directionality solution further doubles the quantity. Unsupervised clustering is therefore used to extract a representative set of sub-flows, on which supervised classifiers are then trained. Naïve Bayes and C4.5 decision trees were selected as the supervised learners, and their effectiveness at distinguishing two time-sensitive applications (an online video game and VoIP) from other traffic with sub-flows was evaluated. Under these conditions, and using 25-packet sub-flows, both algorithms were found to effectively distinguish each traffic application. The C4.5 algorithm achieved 97% precision on the online game and 99.2% precision on VoIP traffic. This outperformed Naïve Bayes' 87% and 95.4% precisions respectively. Both approaches also obtained recalls of approximately 99% for each application. While the sub-flow approach was both novel and shown to be very effective in these experiments, each experiment only aimed to distinguish one temporal application from other applications, i.e. when evaluating one time-sensitive application, the other time-sensitive application was excluded from training and testing datasets. For more general traffic classification tasks such as classifying multiple temporal and non-temporal applications, the suitability of sub-flows was not explored. A general setting poses multiple challenges for the proposed approach, including the difficulty of choosing an appropriate sub-flow length so that all temporal and non-temporal applications can be cleanly separated.

In [20], Bernaille and Teixeira propose the use of supervised clustering to classify encrypted traffic. Having demonstrated the effectiveness of clustering using Gaussian Mixture Models (GMMs) [21], here they expand the approach to classify SSL encrypted TCP connections. As in unsupervised approaches, the clustering is performed without labels. Labelled data is then used to map clusters to applications. In this approach, the authors use incomplete flows to allow earlier detection. Specifically, the statistical features are generated from only the first three packets in a flow. Using these statistical features, flows are classified using a modified nearest cluster classifier with special consideration for encrypted traffic. The classifier first identifies whether or not a new flow is SSL encrypted. If the flow is not encrypted, it is simply classified via its most similar cluster. If it is encrypted, then the system identifies the first three packets of the flow containing application data (as the first encrypted SSL packets conduct the SSL handshake). Then, statistical features of these three encrypted packets are used to identify the application. With this simple consideration for SSL, the authors demonstrated 85% classification accuracy on encrypted flows. However, this approach had a number of drawbacks. First, while statistical features were used, port numbers were also included as features. This was likely required due to using only three packets per flow to calculate statistics, which would offer less resolution. The authors acknowledge the difficulty in finding a considerable amount of non-SSL traffic using standard SSL ports, further demonstrating the unreliability of using port numbers. Furthermore, the approach was very specific to SSL encryption, with no other encryption mechanisms being supported or considered here.

Fahad et al. also proposed a clustering-based supervised flow classification approach, called CluClas [22]. In this model, *k*-Means clustering is applied as a pre-processing step to the reduce noise and quantity of their traffic flow data. The flow at the centroid of each cluster is kept, and each centroid is used as a representative sample to train and classify with Hidden Markov Models (HMMs). A single HMM is built for each traffic class. Using the set of HMMs, new flows can then be classified into the traffic class for which the log-likelihood is highest. Overall, CluClas was shown to give up to 97% accuracy when using a sufficiently large number of clusters in the pre-processing stage. Effectiveness was seen to increase as the number of clusters increased, with the reported best results being obtained using $k = 400$ clusters. Compared against pure *k*-Means and HMMs, Fahad et al. found the proposed CluClas system gave consistent improvement in effectiveness across four datasets. Accuracy improved by 7.19% to 17.48%, and F-measure improved from 3.47% to 12.21%. The key contribution of this method is the demonstrated strength of using clustering as a pre-processing step. The expected purity of flow clusters allowed a significant reduction in noise when training the HMMs. While slower to train and classify than *k*-Means, CluClas was shown to be more efficient than HMMs in a scalability

analysis. However, not only does this approach suffer from the requirement for fully labelled training data, the use of *k*-Means naturally means the approach requires an appropriate number of clusters to be chosen. Using too few clusters was shown to cause weaker overall performance, as the less pure clusters results in HMMs being trained on unrepresentative flow sets. Furthermore, despite using clustering, unknown applications are not supported.

The range of approaches described above demonstrate how even basic supervised approaches can excel at the traffic classification problem. Furthermore, the strength of supervised approaches has allowed more complex and unique traffic classification problems to be considered. Despite the effectiveness, the requirement of pre-labelling the large volumes of data generated by networks raises concerns for the practicality of all supervised approaches [15]. Additionally, the approaches have a natural inability to support unknown traffic classes, as supervised learners can only classify into classes observed during training. Semi-supervised approaches described in the following section aim to counter these drawbacks.

2.3 Semi-Supervised Approaches

Combining the strengths of unsupervised and supervised approaches, semi-supervised approaches have become increasingly popular for traffic flow classification. These systems are motivated by the realistic use of minimal portions of labelled training data to solve the application identification problem of unsupervised clustering approaches, and to identify unknown classes [6], [8].

An early semi-supervised network traffic classification approach by Qian et al. [23] classified traffic applications with GMMs. Using maximum a posteriori estimation to deduce their GMM parameters, a notable advantage of the approach is the automatic determination of the number of clusters. After clustering, any pre-labelled flows in a cluster are used to automatically map clusters to applications. The authors use a comprehensive set of 248 features per flow, with 10% of the flows pre-labelled. The majority of the features are statistical; however, some non-statistical features, including port numbers, appear. This feature set is notably large compared to related classification approaches, and its size resulted in feature selection being an integral component of this system. Interestingly, the authors report that feature selection resulted in between just two and five features per dataset. The exact feature subsets were not disclosed, but this indicates that certain features can be extremely strong discriminators of traffic classes. Between 80% and 94% classification accuracy was reported across 10 datasets. The important contribution of this foundational semi-supervised work was therefore a clear viability of using primarily unlabelled data to solve the cluster identification problem of unsupervised clustering. However, the approach was not evaluated against other models. Additionally, while 10%

pre-labelled data is a considerable improvement over supervised approaches, in the context of real-world networks, this still corresponds to very large number of flows. Due to the relatively large portion of labelled flows, unknown flows were also not considered in this approach.

Erman et al. proposed a semi-supervised traffic classification system in [6] with a key focus on unknown flow support. While their use *k*-Means clustering on statistical flow features was reminiscent of unsupervised approaches, the primary innovation of this work was the use of probabilistic assignment for known and unknown application identification. As in [23], each cluster is automatically mapped to a traffic application by identifying its most common labelled flow. The simple yet critical difference in this system is that clusters without any labelled instances are mapped to the "unknown" application class. The automatic separation and identification of new and unknown applications is a significant advantage over comparable unsupervised and supervised approach. In supervised approaches, unknown applications flows are invariably, and incorrectly, assigned to known application classes. Meanwhile, in comparable unsupervised approaches, the identification of unknown classes requires manually analysing each cluster. The small amount of labelled data thus automatically mapped clusters to known applications and identified unknown classes. Using up to 8,000 pre-labelled flows (in a dataset in excess of 900,000 flows), Erman et al. demonstrated above 90% classification accuracy. At fewer than 1% pre-labelled training data, this system can be considered significantly more realistic than the 10% used in [23]. Furthermore, precision remained high even when fewer labels were available. However, poor recall is the significant weakness of the approach. When the pre-labelled portion of flows is insufficient, many known-class clusters are incorrectly identified as unknown applications due to a lack of pre-labelled members.

This approach is improved in [8], where Zhang et al. automatically extend an initially small, labelled set of flows using unique flow properties, therefore minimising clusters without labels. This is achieved using “flow label propagation”. The domain-specific technique first groups correlated flows, i.e. flows sharing the same destination IP address, destination port number, and protocol. Given that two correlated flows are expected to originate from the same traffic class, any labelled flow can then automatically share its class label with all correlated unlabelled flows. The labelled dataset is therefore automatically extended. The value of the approach is demonstrated with approximately 100 pre-labelled flows per known class being shown to be sufficient for effective classification. This number is an extreme reduction compared to earlier semi-supervised approaches, and is therefore much easier to acquire. The proposed system further uses correlated flow properties to apply compound classification. A majority vote classifies all flows into correlated flow groups to correct small cases of classification

error. Through a systemic evaluation, the proposed approach was reported to consistently outperform classic supervised learning algorithms when unknown applications were present. With two unknown classes (DNS and SMTP), the Zhang approach outperformed C4.5 decision trees, *k*-Nearest Neighbours, and Bayesian networks in terms of overall accuracy and F-Measures for each traffic class. Given that supervised classifiers cannot support unknown classes, these results are unsurprising. The evaluation did not consider the case where no unknown classes are present, but we would expect notably improved performance of supervised learners here. The proposed approach was also shown to outperform the Erman approach from [6]. The improvement is primarily attributed to the label propagation technique, which was shown to be able to dramatically increase the size of an initially small, labelled dataset with good reliability. There were therefore significantly fewer clusters without any labels. While we consider the proposed Zhang approach to be a leading semi-supervised approach, the reliable identification of some classes still proved to be a challenge. In particular, the HTTPS and BitTorrent classes performed notably poorly relative to the other classes, the reasons for which were not explored.

A more recent semi-supervised approach combines the unknown flow detection from clustering algorithms with the proven effectiveness of supervised techniques [24]. The system begins by initially clustering the full training dataset. As in previous works, clusters without labelled flows are determined to be clusters of unknown applications. This approach is unique in that it first trains an intermediary supervised classifier on a subset of the complete dataset; the labelled portion combined with flows present in unknown clusters. The intermediary classifier then classifies the complete training dataset to identify unknown flows. A final supervised model is then trained using the labelled and unknown flows identified from this process. Using random forests as the supervised classifier, the effectiveness was shown to be extremely good. Varying between 4,000 and 20,000 pre-labelled flows, the results of the proposed system were reported to be consistently above the semi-supervised classifier in [8] in terms of accuracy and F-measures. However, this approach required significantly more labels to reliably identify unknown class samples. Too-few labels would result in erroneous selection of unknown class samples and result in a poor final classifier. The system is also notably complex, requiring three models to train a classifier. Despite the drawbacks, the unique approach of combining the high performance of supervised learners with unknown flow support represents the main contribution of this work.

2.4 Alternative Flow Formats and Feature Sets

The majority of traffic flow-based classification approaches we encountered use similar sets of statistical features. Due to the lack of a standard definition of network flows, equivalent feature sets are not derivable for every flow format used. In this section, we briefly explore some works using alternative flow formats. In particular, we identify works using the NetFlow [25] flow format.

In [26], Soysal & Schmidt apply classic supervised learning algorithms to NetFlow data for flow classification. The NetFlow format flow dataset resulted in a limited statistical feature set, thus the authors opted to include some non-statistical features. Notably, source port and destination port were used as features. Both decision trees and Bayesian networks were found to be extremely effective, with upwards of 97% accuracy for both approaches. The key finding in this work was that the source port was a dominating feature in classification. When P2P flows masqueraded on port 80 (the standard port for HTTP), both supervised algorithms had moderate decreases in accuracy for both the P2P and HTTP classes. Compared to the ideal setting, the Bayesian networks showed 5.3% lower recall of P2P, while decision trees were 1% lower. However, the supervised learners in these evaluations were specifically trained on examples of P2P using port 80. Therefore if P2P flows were to use any other non-standard port, the impact on effectiveness is expected to be much more notable. In the case of simulated dynamic port number allocation, the effectiveness loss was more significant. The accuracy of Bayesian Networks in particular reduced by over 16% in this setting. The impact on decision trees was less drastic, but again the supervised learners were trained specifically to expect certain port ranges for applications using dynamic allocation. We consider this an ideal scenario that is not expected to generalise to real-world settings. Therefore, while both supervised learners were shown to yield clear improvements over pure port-based classifiers, being trained with ports had clear impacts on generalizability.

A number similar works have attempted traffic classification on the same kind of flow records. The authors of [27] evaluate a hybrid signature-based and machine learning approach. Flows successfully identified with the DPI tool are used to train a supervised classifier using NetFlow features like in [26]. Flows that cannot be identified with DPI are identified with the supervised classifier. Similar flow features are used again in [28], where PGMs are applied in a semi-supervised manner to NetFlow records. The uniqueness of this approach is the inclusion of flow start timestamps, end timestamps, and IP addresses, as features. Averaging classification accuracy above 90% effectiveness, the model's success is partially attributed to using both ports and IP addresses as features. Again, using NetFlow data, supervised algorithms are used to detect brute force attacks on the SSH protocol in [29]. While not application classification, attack detection is a closely related problem, and the findings here are

relevant for application classification. Notably, the authors investigate including ports as features alongside the small statistical feature set. The important finding was that port numbers could be overwhelmingly dominant features. It was found that using ports as features causes decision trees to largely ignore statistical features, and port number inclusive models were very specific to the training data. It was concluded that port numbers improves the performance of attack detection at the cost of generalizability.

The commonality across all of these works is therefore use of port numbers, and the resultant loss of generalizability. These features make the classifier extremely specific to a single network and susceptible to non-standard port numbers. The classifiers would likely need to regularly be re-trained on large quantities of labelled data to remain effective. Meanwhile, classifying traffic using exclusively statistical features from this flow format has not been thoroughly investigated in the literature.

Chapter 3
Datasets

Four datasets were acquired and used throughout this thesis. Table 3.1 outlines the key characteristics of each dataset. In-depth descriptions of each dataset, including their sources and processing, are given in sections 3.1–3.3.

Table 3.1: Summary of datasets used in the experiments.

Name	Format	Public	Bidirectional	Features	Classes	Flows
wide	Packets	Yes	Yes	39	7	525,603
waikato	Packets	Yes	Yes	39	6	594,786
NetFlow	NetFlow	No	No	7	7	412,335
bidirectional-NetFlow	Aggregated NetFlow	No	Yes	10	7	382,755

3.1 *wide* Dataset

Our first dataset originates from the WIDE network traffic archive [30]. The WIDE traffic archive is a collection of publicly available, anonymised packet header traces collected from a link that connects WIDE to its upstream ISP. Data from this archive is commonly used for evaluation [8], [22], [24]. Our sample originates from a 10-hour traffic capture, captured in March 2008 at samplepoint-F on the WIDE network. NetMate [31] is used to convert packets into bidirectional flows and compute 39 purely statistical features to create our *wide* dataset. The full set of features is described in Table 3.2. Approximately 525,000 flows are obtained across the traffic trace's seven most common application classes. The seven classes are: HTTP, HTTPS, BitTorrent, DNS, SMTP, FTP, and SSH. The dominant classes are HTTP and DNS. For ground truth labelling, standard port numbers are used to automatically identify classes. While we acknowledge identifying ground truth classes with this method will introduce error, this is a common labelling approach used in the literature and the error introduced is expected to be small [18].

Table 3.2: Set of statistical features in the *wide* and *waikato* datasets.

Name	Description	Features
Packets	Number of packets transferred in each direction.	2
Bytes	Total number of bytes transferred in each direction.	2
Packet Size	Minimum, mean, maximum, and standard deviation of bytes transferred in each direction.	8
Packet Time	Minimum, mean, maximum, and standard deviation of time between packets in each direction.	8
Duration	Total time the flow was alive.	1
Active/Idle Time	Minimum, mean, maximum, and standard deviation of time a flow was idle/active.	8
Subflow Features	Number of bytes and packets in a sub flow in each direction.	4
Flags	Number of times the PSH/URG flag was seen in each direction.	4
Headers	Total bytes used for headers in each direction.	2

3.2 *waikato* Dataset

The *waikato* dataset is a sample of the publicly-available Waikato VIII traffic traces [32]. The complete Waikato VIII dataset comprises a series of packet header traces captured between the University of Waikato network and the Internet. Our dataset is sampled from a capture between September 30th 2011 and October 1st 2011. The data is first converted from ERF format to PCAP trace file format using the libtrace tool [33], before being converted into statistical features using the same method as the previous dataset. The derived features for this dataset are equivalent to those in the *wide* dataset (Table 3.1). Ground truth labelling was obtained using the same method as the previous datasets. The six major classes comprising this dataset are: HTTP, HTTPS, NTP, DNS, SMTP, and SSH. The dominant class is HTTP.

3.3 NetFlow and bidirectional-NetFlow Datasets

These datasets were created from a data sample supplied by the Australian Academic and Research Network (AARNet) [35] for research purposes. This dataset was captured from an IPFIX [34] flow stream over a 5-minute period on June 27th, 2016. Despite the small capture window, the large scale of the network offered a sufficient sample. Over 410,000 flows are sampled from the capture, corresponding to the same seven classes as in the *wide* dataset. The flows are converted to NetFlow v9 format and anonymised using nfdump [36] to create our *NetFlow* dataset. The dominant classes in this capture are HTTP and HTTPS, together accounting for approximately 60% of the records. The *NetFlow* set, described in Table 3.3, contains eight statistical features equivalent to those found in standard, unidirectional NetFlow flows.

Table 3.3: Set of features in the *NetFlow* dataset.

Name	Description	Features
duration	Total time the flow was alive	1
flags	Bitwise AND of TCP flags	1
tos	Type of service	1
packets	Number of packets in the flow	1
bytes	Number of bytes in the flow	1
pps	Number of packets per second in the flow	1
bps	Bits per second	1
Bpp	Mean number of bytes per packet	1

The second feature set (Table 3.4), contains statistical features for bidirectional flows. As the original flows were unidirectional, bidirectional flows are creating using nfdump to aggregate flows using IP addresses, ports, and protocols. Due to the aggregation, this process excludes the *flags* and *tos* features

from the *NetFlow* feature set, but additional statistical features are derived. Ground truth class labels for both sets were obtained as described in 3.1.

Table 3.4: Set of features in the *bidirectional-NetFlow* dataset.

Name	Description	Features
duration	Total time the flow was alive	1
packets	Number of packets transferred in each direction	2
bytes	Number of bytes transferred in each direction	2
bps	Mean number of bits per second transferred while the flow was alive	1
pps	Mean number of packets per second transferred while the flow was alive	1
Bpp	Mean number of bytes per packet in each direction	2
Flows	Number of flows aggregated into this bidirectional record	1

Chapter 4
Exploratory Comparative Analysis of Network Flow Formats for Traffic Classification

In this chapter, an exploratory analysis is conducted to evaluate the strength of a flow format not commonly used for the traffic classification problem in the literature. This flow format is the NetFlow format [25]. In terms of purely statistical features, we compare NetFlow flows against the format more typically used across the literature. Furthermore, we analyse a bidirectional NetFlow feature set. To the best of our knowledge, it is the first time purely statistical NetFlow features and bidirectional NetFlow features are considered for traffic classification. Throughout this chapter we combine visualisation, feature selection, and supervised learning techniques for a comprehensive comparative evaluation.

4.1 Motivation

A traffic flow is generally defined as connection between two hosts, sharing the same 5-tuple: source IP, source port, destination IP, destination port, and protocol. This definition is high-level, with the specifics often left to the implementation. Therefore we cannot expect that all network flows are equivalent. However, the wide range of approaches dedicated to classifying network flows primarily consider detailed statistical feature sets derived from packet-level traffic traces [11]-[16], [18]-[24]. Many real-world networks capture traffic directly at the flow level, with original packets being discarded. In such formats, the detailed statistical features like in our *wide* dataset (section 3.1) are not readily available. However, despite being valid network flows, little consideration has been given to these other formats in existing flow classification approaches. It is therefore unclear how existing machine learning approaches can classify some legitimate kinds of flows using purely statistical features. Here, we focus on NetFlow-like flows as described in section 3.3. Existing works that use NetFlow data [26]-[28] report good effectiveness, but rely on port numbers as features. Ports are considered unreliable [4]-[5], [26] yet very discriminative features due to the expected variation [29]. We consider the use of port numbers to be a key weakness of these approaches, which removes generalizability from classifiers and introduces the same drawback of basic port number classifiers.

We choose to focus on NetFlow and NetFlow-like flows purely for the prevalence of the format. Being built into many Cisco network products, NetFlow is an extremely common flow format. A 2013 survey [37] found NetFlow to be the most common flow capture in existing networks. While multiple versions of NetFlow exist, v5 and v9 are the most popular [38]. The static flow format of NetFlow v5 is currently the most common [39], meaning that the detailed features available in our *wide* flows are not always readily available. The newer NetFlow v9 is more flexible, but the default capture fields are

largely equivalent to that of v5 and hence exclude many *wide* fields by default. Capturing extra fields requires both work and justification in the face of big data challenges. That is, due to the volume and velocity of network traffic, capturing even a few unnecessary attributes can significantly increase the size of datasets.

Through an exploratory analysis, the work in this chapter is motivated by the importance of understanding the viability of using common NetFlow flows for traffic classification. To demonstrate this importance, we use a real traffic capture provided by a major Internet service provider who was interested in understanding how machine learning techniques could be applied to their data. Due to the proven effectiveness of the features found in our *wide* dataset, a key component of our analysis is to understand relative strength.

4.2 Problem Statement

We are given two datasets of fully labelled network flows $F1 = \{x_i \mid i = 1, 2, \ldots n\}$ and $F2 = \{y_j \mid j = 1, 2, \ldots m\}$, where $n \approx m$ and $F1$ and $F2$ are different flow formats. The flows in each dataset are described by a set of statistical features S, where $F1_S \neq F2_S$. Each flow was generated by some traffic class c from a set of classes C. If C_{F1} describes the set of traffic classes in $F1$ and C_{F2} describes the set in $F2$, then $C_{F1} \cap C_{F2} = C_{F1} = C_{F2}$. From $F1$ and $F2$, we aim to identify the more suitable flow format for statistical network traffic classification. We further aim to identify features causing the suitability to better understand the domain.

4.3 Approach

To identify the comparative strengths of each flow format, we use two datasets. The *wide* dataset (described in 3.1) is used to represent flows used in existing machine learning approaches. We use the two NetFlow feature sets described in section 3.3 for both our *NetFlow* and *bidirectional-NetFlow* flows. As explained in section 3.3, the aggregation of unidirectional NetFlow records allows additional directional features, which are more similar to those in the *wide* set albeit still with some differences. When comparing flow formats, we consider it ideal to capture each format simultaneously on the same network. While we are not aware of any datasets that satisfy this ideality, our datasets are captured on real-world networks and contain flows that we expect to be representative of each traffic class. Furthermore, the same seven classes are available in both datasets in similar proportions. We thus assert it is reasonable to compare these datasets at a high-level. It is worth noting that not all approaches in the literature use identical feature sets. However, our *wide* dataset covers the features most commonly observed in state-of-the-art approaches. Thus we consider these flows sufficiently representative.

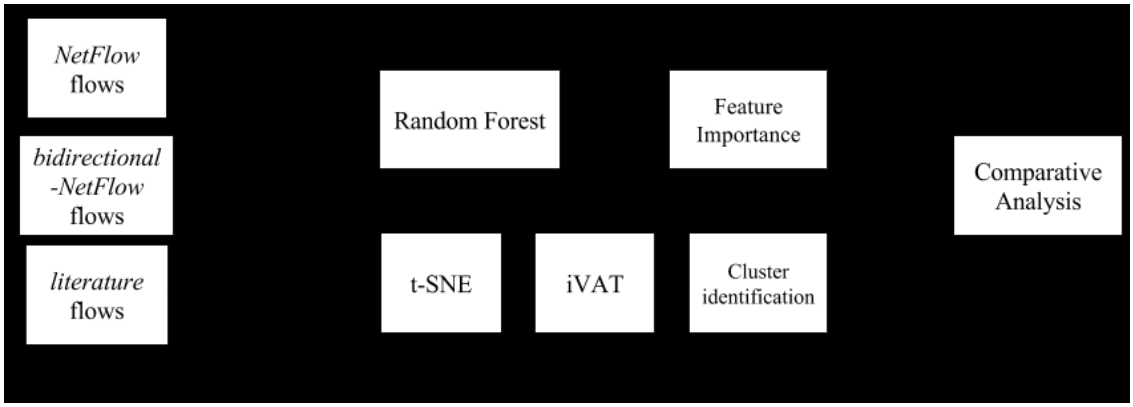

Fig. 4.1. Systematic approach to comparative visual analysis between flow formats.

The overall exploratory approach is described in Fig. 4.1. For each set of flows, two complimentary modules contribute to our analysis. In the supervised learning module, overall classification effectiveness is evaluated in an ideal, supervised setting. Random forests are selected as the supervised classifier due to their proven effectiveness for traffic classification [18]-[19]. The requirement for fully labelled data can be considered unrealistic in real-world settings. However, supervised classifiers are also expected to offer the highest performance when sufficient labelled data is available. Thus, we use random forests to indicate upper-bound effectiveness. Furthermore, we use the supervised algorithm for its implicit ability to rank features and indicate strong discriminators.

The supervised learning results are contextualised by the visualisation module. In this component, t-Distributed Stochastic Neighbour Embedding (t-SNE) is used to visualise each dataset. The technique is selected for its strong visualisations, which generally avoids crowding data points towards the centre of the visual space [40]. Previous work has demonstrated that features in the *wide* dataset have a requirement for many more clusters than classes [3], [13], [16]. To identify the cluster tendency of each flow format, we therefore apply the iVAT algorithm to each t-SNE representation. iVAT uses a reordering technique to group similar data points and visually indicate clusters. Combining the reordering and visualisation, we manually highlight the iVAT clustering on the t-SNE visualisation. The result of combining t-SNE and iVAT is a clear visual indication of the underlying structure of the data.

Our comparative analysis approach is therefore to (1) evaluate overall effectiveness for each flow format in an ideal setting, and (2) explain the comparative effectiveness using feature ranks and underlying structure identified in each visualisation. The techniques used in our approach are described in more detail below.

4.3.1 t-Distributed Stochastic Neighbour Embedding

t-SNE [40] is a non-linear dimensionality reduction technique for the visualisation of high-dimensional data. t-SNE falls into a class of techniques that assumes that high-dimensional data exists on a lower-dimensional manifold. Following this assumption, these techniques aim to preserve the structure of the high-dimensional data in a two or three-dimensional representation, thus allowing visualisation.

The overall approach for obtaining low-dimensional representations is as follows. First, Euclidean distances are calculated between each pair of objects x_i and x_j in the high-dimensional data, X. On each pair of objects, a conditional probability $p_{j|i}$ is calculated using Equation 1. This probability $p_{j|i}$ represents the likelihood that, for some point x_i, its neighbour is x_j under a Gaussian distribution centred at x_i.

$$p_{i|j} = \frac{exp(-||x_i - x_j||^2 \div 2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2 \div 2\sigma_i^2)} \tag{1}$$

For each of these Gaussian distributions centred at some x_i, a standard deviation σ_i is required for the calculation. Each σ_i is determined by finding the value that results in a probability distribution over the data P_i matching a user-defined perplexity. The perplexity, $Perp$, is a measure of information calculated using the Shannon entropy (Equation 2), as given in Equation 3. However, it is worth noting that t-SNE is robust to variations in this parameter.

$$H(P_i) = -\sum_j p_{i|j} \, log_2 p_{j|i} \tag{2}$$

$$Perp(P_i) = 2^{H(P_i)} \tag{3}$$

More similar pairs of data points therefore have relatively higher probabilities $p_{j|i}$. This is then used to define the joint probabilities p_{ij} in Equation 4. The result is a joint probability distribution P over the high-dimensional space.

$$p_{ij} = \frac{p_{(j|i)} + p_{(i|j)}}{2n} \tag{4}$$

A similar process is repeated to obtain a joint probability distribution Q in the low-dimensional representation of the space. Using pairs of points y_i and y_j from the low-dimensional space, t-SNE employs a Student-t distribution with one degree of freedom. Such a Student-t distribution is used to allow heavy-tailed distributions, therefore reducing crowding around the centre of the visualisation. That is, heavy tails allows dissimilar data points to be better distanced in the low-dimensional space. From this distribution, the low-dimensional joint probabilities q_{ij} are calculated per Equation 5.

$$q_{ij} = \frac{(1+||y_i - y_j||)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||)^{-1}} \quad (5)$$

t-SNE finally aims to minimise the Kullback-Leibler (KL) divergence between P and Q. The KL divergence describes how effectively the high-dimension joint distribution P is modelled by the low-dimension joint distribution Q. A minimal divergence between Q and P can be found through gradient descent using the cost function C given in Equation 6.

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \quad (6)$$

The gradient of the KL divergence between P and Q can be derived using Equation 7, with positive gradients representing attraction between points. Negative gradients represent repulsion. Gradient descent thus iteratively works towards a minimum divergence to select and plot the set of low-dimensional set of points Y that best represents the underlying manifold of the high-dimensional data.

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \quad (7)$$

With both computational and memory complexities of $O(n^2)$, it can be somewhat impractical to apply t-SNE on large datasets. In these works, consideration was given to other dimensionality reduction techniques such as Multidimensional Scaling and Primary Component Analysis. However, early results indicated that each of our datasets was extremely susceptible to crowding, which t-SNE is designed to avoid. We therefore determined t-SNE to be the most appropriate technique.

4.3.2 Improved Visual Assessment of Cluster Tendency

The improved Visual Assessment of cluster Tendency (iVAT) algorithm [41] belongs to the VAT family of visualisation algorithms, which indicate the natural number of clusters k in a dataset. The value of such methods is clear when mainstream clustering algorithms such as k-Means parameterize the number of clusters. The VAT and iVAT techniques work as described below.

A pairwise dissimilarity matrix $\mathbf{D} = [d_{ij}] \in \mathbb{R}^{n \times n}$ is calculated on dataset $X = \{x_1, x_2, \ldots x_n\}$, such that the dissimilarity of any pair of objects x_i, x_j is found in $\mathbf{D}$ at position d_{ij}. VAT aims to find a matrix $\mathbf{P}$ by which to reorder $\mathbf{D}$, such that the resultant matrix $\widetilde{\mathbf{D}}$ aligns blocks of similar objects diagonally. When the reordered matrix is displayed as a grey scale image, these blocks correspond to cluster tendency in terms of both cluster size and number. Equation 8 gives the original VAT formulation.

$$\widetilde{\mathbf{D}} = \mathbf{P^T D P} \quad (8)$$

iVAT uses the same reordering process, instead using a path-based dissimilarity matrix, **D'**. Therefore even if the distance between two individual points is large, they are still considered similar if there is a path of similar objects connecting them. This consideration allows the capture of unusually shaped clusters, and clusters with less distinct boundaries. **D'** is obtained by considering **D** as a fully-connected graph, with each object acting as a graph node and each edge between object pairs x_i and x_j being weighted by their dissimilarity, d_{ij}. Their dissimilarity in **D'** is thus d'_{ij}, the path-based distance between the points (Equation 9). Equation 9 dictates that the dissimilarity between x_i and x_j along each possible path l of length $|l|$ connecting the two objects is the maximum dissimilarity between any two adjacent points in this path, x_h and x_{h+1}. Therefore d'_{ij}, is the minimum of these dissimilarities.

$$d'_{ij} = min_{l \in L_{ij}}(max_{1 \leq h \leq |l|}(d_{h,h+1})) \quad (9)$$

From this it follows that **D'** = $[d'_{ij}]$, and the iVAT reordered matrix can be calculated as in Equation 8. From this the image is displayed and interpreted as in classic VAT.

With $O(n^3)$ complexity, it can be largely infeasible to apply iVAT on large datasets. Although reformulations [42] have shown that this is reducible to $O(n^2)$, this still restricts the quantity of data. bigVAT [43] and clusiVAT [44] are examples of related algorithms designed to accommodate more data. However, iVAT is chosen for this analysis, primarily for its strength at visualising data with unclear cluster boundaries. As previous clustering work on the traffic classes has consistently demonstrated [3], [13], [16], for a dataset with c classes, a number of clusters $k >> c$ is required. Due to the large disparity between k and c, clusters indicated by these algorithms are not expected to be crisp in this domain. iVAT is thus selected for its more robust visualisations.

4.3.3 Random Forests and Feature Importance

Decision trees are predictive models that map observations to a tree-like structure, with leaves as class labels and conditions on branches dictating how to traverse the structure depending on the feature values observed. We use the Extra Trees Classifier from [45], an implementation of random forests (Algorithm 4.1). This is a supervised ensemble learning method, which employs a bagging strategy to train and combine numerous decision trees into a meta-classifier. New instances can thus be classified as the most common guess from all trees in the forest.

We utilize random forests in our analysis for both supervised classification and feature selection. Feature selection techniques are techniques that aim to reduce an initial feature set to a core subset that excludes redundant and irrelevant features. The intuition is that removing poor feature can result in both simplified and strengthened models. Decision trees implicitly perform feature selection by

splitting at leaves depending on the discriminative power of the available features. The degree to which a feature reduces the decision tree splitting criteria thus correlates with its discriminative power.

```
Input: number of trees T; labelled dataset X; number of features N
Result: meta-classifier Forest
Forest = []
for t = 1 ← T:
   get dataset sample Y such that Y ⊂ X,
          get subset of features F such that |F| ≤ N
   train decision tree DT_t using Y, F
   internally evaluate DT_t on Z, where Z ⊂ X, Y ∩ Z = Ø
   add DT_t to Forest
end
Classify: Given data point y, y_class = mode([DT(y) for DT in Forest])
```

Algorithm 4.1. Random Forests

Numerous metrics have been proposed to formalise the strength of features. In this chapter, we opt to use Gini importances to identify and compare valuable features in each flow format. The Gini importance of a feature is a measure of how it reduces the decision tree criteria [46]. Given a random forest of T decision trees, features can therefore be ranked by their Gini importance [47]. Other measures such as entropy can similarly be used, but these are largely equivalent [48]. The Gini importance is calculated using Gini impurity, a measure of the distribution of labels in a dataset. Consider a node n in a single decision tree. Given C classes in the data, the probability of the data available at n being of some class c is p_i. The Gini impurity at n can be calculated with Equation 10.

$$G(n) = \sum_{i=1}^{C} p_i(1 - p_i) \tag{10}$$

Each time a split occurs in the decision tree, the Gini impurity of the children nodes is strictly smaller than that of the parent node due to the reduction in data. From this, the Gini importance for each feature can be calculated by summing the decrease in Gini impurity at each node across the trees in the forest [49]. Let $L(n, t_i)$ equal the loss in Gini impurity between node n and its two children in decision tree t_i. Equation 11 gives the Gini importance I_G for each feature f in a random forest of T trees.

$$I_G(f) = \sum_{i=1}^{T} \sum_{n=1}^{N} L(n, t_i) \tag{11}$$

4.4 Experimental Evaluation and Results

In this section, the classification effectiveness, feature strengths, and visualisations for each flow format are presented. Due to the relatively demanding complexities of both t-SNE and iVAT, we were required to limit the amount of data used in the visualisation module. In this module we thus used a random balanced sample of 7,000 flows, 1,000 per traffic class. We acknowledge using a sample of the dataset potentially neglects valuable information from unused flows. However, we will demonstrate

that our sample is sufficient to draw important conclusions about the overall structure of the each dataset. Additionally, supervised learning is conducted on the complete datasets to support the visualisations. We train random forests of 10 trees, which are internally evaluated using out-of-bag samples. Effectiveness is thus calculated by evaluating each decision tree in the forest on data excluded from its training sample. This is shown to be an effective method of evaluating random forests [50] by allowing training and testing on the full dataset without risking overfitting.

4.4.1 Evaluation Metrics

In evaluating the effectiveness of machine learning approaches, we use two standard metrics. The first is accuracy, i.e. the number of correctly classified flows out of all classifications made. This metric is used to evaluate overall classifier performance.

$$Accuracy = \frac{(Correctly\ Classified\ Flows)}{(Total\ Number\ of\ Flows)} \tag{12}$$

The second metric used is F-measure, i.e. the weighted harmonic mean of precision and recall. Precision is defined as the ratio of flows correctly classified as a class to all flows classified as that class. Recall is defined as the ratio of flows correctly classified as some class to all flows truly belonging to that class.

$$F\text{-}Measure = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \tag{13}$$

Throughout this thesis, the F-Measure is used to evaluate the performance for each class individually.

4.4.2 Classification Effectiveness

The overall accuracy of a random forest trained on each flow format can be seen in Figure 4.2(A).

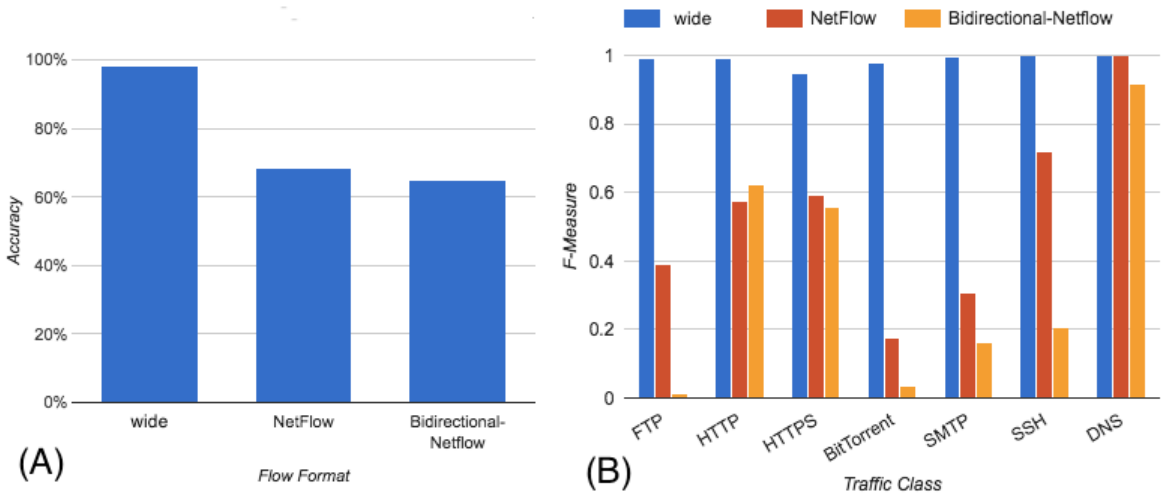

Fig. 4.2. (A) Overall classification accuracy for each flow format. (B) F-measure for each traffic class in each flow format.

The overall accuracy from the *wide* flows was extremely good at 98.35%. The *NetFlow* flows performed approximately 30% poorer, with just 68.59% overall accuracy. The weakest performance came from the *bidirectional-NetFlow* dataset. At 65.02% accuracy, this dataset was more comparable to the *NetFlow* flows. The F-Measures presented in Figure 4.2(B) explain the source of the differences in accuracy. The F-Measures for the *wide* flows were most consistent with at least 0.95 for all classes. In both NetFlow datasets, the F-Measures had significantly more variation. All datasets were comparable for the DNS class. In this class, both *NetFlow* and *wide* had F-Measures of 1.00, indicating near perfect ability to distinguish DNS flows from other classes in both datasets. The *bidirectional-NetFlow* feature set also performed relatively well at 0.92. For the remaining classes, the *wide* flows significantly outperformed both *NetFlow* and *bidirectional-NetFlow*. The *NetFlow* class performed reasonably well for SSH with an F-Measure of 0.71, but is notably lower than the *wide* set's score of 1.00. The *bidirectional-NetFlow* was relatively poor at just 0.21 for the same class. The remainder of the *NetFlow* F-Measures were below 0.60, with the worst being BitTorrent with a score of just 0.18. *bidirectional-NetFlow* tended to perform even worse than *NetFlow*, with the exception of HTTP. At 0.62, the F-Measure for *bidirectional-NetFlow* was 0.05 higher than *NetFlow*. The score for this class was above 0.99 for the *wide* flows. With an F-Measure of 0.95, w*ide*'s poorest performing class was HTTPS. Again this was significantly above the *NetFlow* and *bidirectional-NetFlow* sets, which scored just 0.59 and 0.56 respectively. While both NetFlow datasets performed poorly on FTP, *bidirectional-NetFlow* was especially poor with an F-Measure of just 0.01.

The feature importance rankings and visualisations hint at the reasons behind the discrepancies in performance, which we further analyse and discuss later in section 4.5.

4.4.2 Feature Importances

In tables 4.1 through to 4.3, the features in each dataset are ranked by their relative importance for predicting each class. As a relative measure, the Gini Importance scores are used for internal comparisons and are not used in comparing each dataset.

Table 4.1: Feature importances for each feature in the *NetFlow* dataset.

Rank	Feature	Direction	Gini Importance
1	flags	Forwards	0.494
2	Bytes per packet (Bpp)	Forwards	0.190
3	bytes	Forwards	0.094
4	duration	Forwards	0.080
5	bits per second (bps)	Forwards	0.077
6	packets per second (pps)	Forwards	0.028
7	packets	Forwards	0.025
8	tos	Forwards	0.012

In the *NetFlow* dataset (Table 4.1), the *flags* feature is seen to be overwhelming considered the most important feature, with more than twice the importance of the second ranked feature; *Bpp*. *Bpp* is still a relatively strong feature, with its score still highly above the remaining features. The remaining six features have notable lower importance scores. The *bytes*, *duration*, and *bps* features are mid-range discriminators with reasonably equal scores, followed by the *pps* and *packets* features, which have a clear decrease in importance. The *tos* feature is clearly weakest at under half the previous feature.

Table 4.2: Feature importances for each feature in the *bidirectional-NetFlow* dataset.

Rank	Feature	Direction	Gini Importance
1	Bytes per packet (Bpp)	Backwards	0.359
2	bytes	Backwards	0.116
3	Bytes per packet (Bpp)	Forwards	0.094
4	duration	–	0.086
5	flows	–	0.085
6	packets per second (pps)	–	0.085
7	bits per second (bps)	–	0.081
8	bytes	Forwards	0.038
9	packets	Backwards	0.035
10	packets	Forwards	0.020

Despite the extra features in *Bidirectional-NetFlow*, the feature rankings presented in Table 4.2 are not dissimilar to those in the *NetFlow* set. Both feature have some clear strongest features with a balanced midrange. In *Bidirectional-NetFlow*, both *Bpp* features are highly ranked, with the backwards direction feature being the strongest in the dataset by a significant margin. Nearby in the third rank, the forwards direction *Bpp* is also reasonably strong. This aligns with the original *NetFlow* set, where the unidirectional *Bpp* feature was second only to *NetFlow*'s unique *flags* feature. The similarities continue with the *bytes* feature following *Bpp* in both data sets. The backward direction feature was again ranked higher, with the forward direction feature offering notably less value. Furthermore, the *duration* feature also found similar midrange rankings in both datasets. The *flows* feature is unique to the *Bidirectional-NetFlow* set, and is approximately equal with the *duration*, *pps*, and *bps* features with medium scores. Notably, the *pps* feature was of midrange importance here, but was of low-range importance in the *NetFlow* dataset. Finally, the *packets* features were the poorest discriminators.

For brevity, only the top 15 of the 39 features in the *wide* datasets are given below in Table 4.3. The best features are clearly more balanced in terms of importance score than both NetFlow feature sets. Notably, there is no single dominating feature. However, the feature with the most importance – maximum packet size in the backwards direction – still scored clearly above the others. In general, statistics involving maximums tended to be stronger features in *wide*, with mean and standard deviation features also occurring throughout. Only one feature using the minimum statistic appeared in

the top rankings. Five inter-packet time features appear in the list of top features, but packet size features tended to outrank these. Seven of the top *wide* features focus on packet sizes, which is consistent with both NetFlow sets and indicates relative strength for these kinds of features. However, the *NetFlow* and *bidirectional-NetFlow* sets have only one packet size statistic, the mean packet size (*Bpp*). In the *wide* dataset, the equivalent features were generally outranked by maximums and standard deviations. In both the *wide* and *bidirectional-NetFlow* sets, mean packet size in the backward direction was higher ranked than the forward direction.

Table 4.3: Top 15 important features in the *wide* dataset.

Rank	Feature Type	Statistic	Feature Direction	Gini Importance
1	Packet Size	Maximum	Backwards	0.181
2	Inter Packet Time	Maximum	Forwards	0.105
3	Packet Size	Maximum	Forwards	0.089
4	Packet Size	Standard deviation	Forwards	0.073
5	Packet Size	Mean	Backwards	0.064
6	Inter Packet Time	Maximum	Backwards	0.060
7	Packet Size	Standard deviation	Backwards	0.041
8	Active Time	Maximum	–	0.040
9	Packet Size	Minimum	Forwards	0.037
10	Packet Size	Mean	Forwards	0.027
11	PUSH Flag	Yes/No	Forwards	0.027
12	Active Time	Mean	–	0.026
13	Inter Packet Time	Mean	Forwards	0.025
14	Inter Packet Time	Standard deviation	Backwards	0.020
15	Inter Packet Time	Mean	Backwards	0.019

In the top 15 features, 14 are directional. Seven of the best features are seen to be in the forward direction, and six in the backward direction. This is in clear contrast to the unidirectional *NetFlow* flows, which only has features in one direction, and the *Bidirectional-NetFlow*, where only half of its directional features ranked well. The best feature in the *NetFlow* data is *flags*, and it does not have an exact equivalent in the *wide data*. The *wide* set, however, has multiple flag features. At rank 11, only the *PUSH flag* feature appears in the top 15 features here. The remaining equivalent features from the *NetFlow* and *bidirectional-NetFlow* datasets, including flow duration and number of packets, do not appear in the top 15 of the *wide* dataset.

4.4.4 Visualisation of *NetFlow* Flows

In the following visualisations, a different coloured circle represents each traffic class. Black loops represent clusters. Fig. 4.3 presents the visualisation of the *NetFlow* flows.

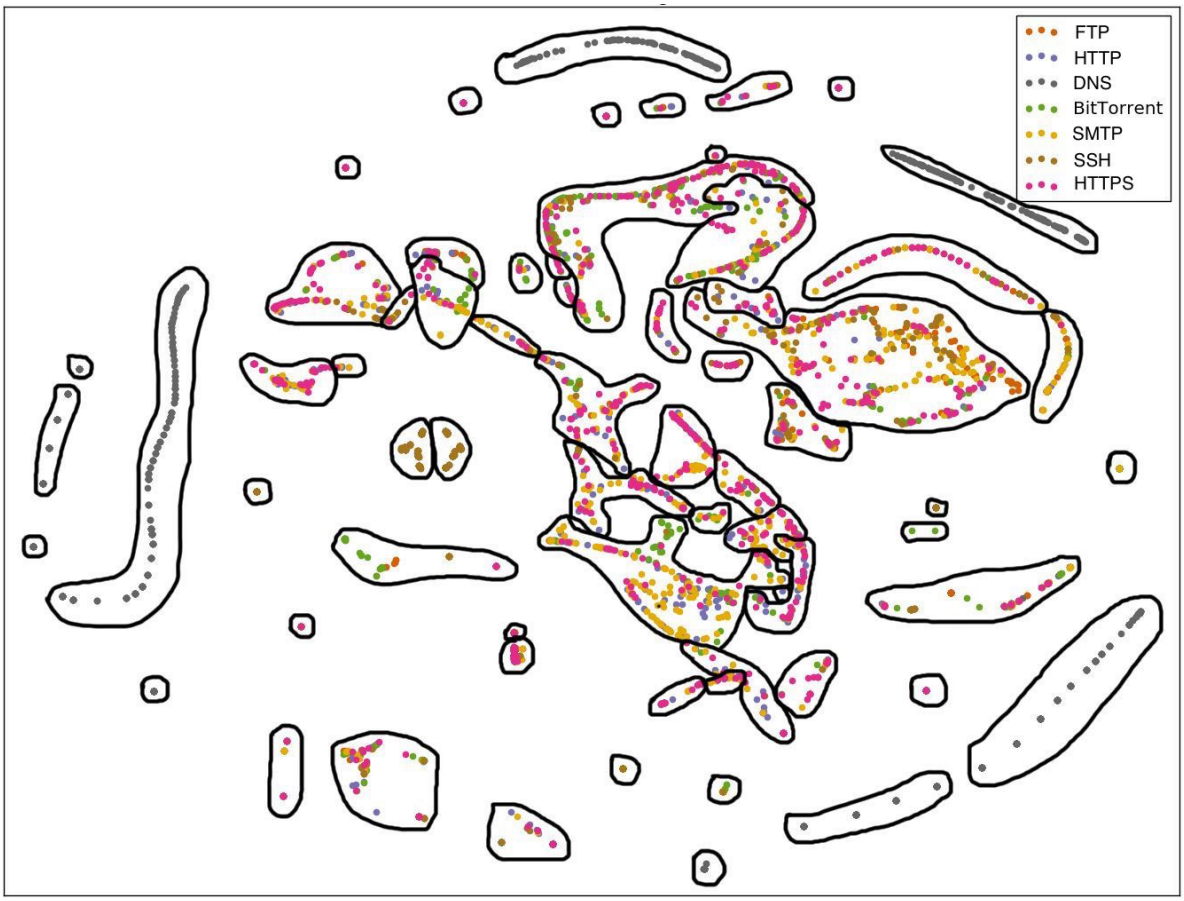

Fig. 4.3. Visualisation of the *NetFlow* dataset.

In this visualisation, many more clusters than classes are identified. This matches expectations given earlier works for effectively clustering traffic flows. A natural consequence is that clusters tend to be quite small. It is notable that in this image, the clusters identified are generally unusual in shape. Towards the centre of the image we find the majority of the flows. Despite the reasonably small clusters, considerable overlap between multiple classes is extremely evident. Furthermore, there is typically limited space between clusters, indicating unclear boundaries. Overall, the visualisation indicates poor clustering.

The exceptions to this are the distinct "chain" shaped clusters appearing on the outskirts of the visual space. The majority of these appear to be purely DNS flows. Furthermore, they are clearly and cleanly separated from other clusters. Some other pure clusters also appear throughout the image. For

example, there is a small cluster of purely SSH flows near the left-most chain. In Table 4.4 and Figure 4.4, we identify some flows comprising these and similar pure structures.

Table 4.4: Table of *NetFlow* flows highlighted in Fig. 4.4. For each flow, a label corresponding to the visualisation is given, as well as the true class and the values of its eight features.

Flow	Class	duration	flags	tos	packets	bytes	pps	bps	Bpp
D1	DNS	0.000	0	0	1	143	0	0	143
D2	DNS	0.000	0	0	1	120	0	0	120
D3	DNS	0.000	0	0	1	89	0	0	89
D4	DNS	0.000	0	0	1	84	0	0	84
D5	DNS	0.000	0	0	1	80	0	0	80
S1	SSH	0.000	2	0	1	40	0	0	40
S2	SSH	0.000	2	0	1	40	0	0	40
S3	SSH	0.000	2	0	1	40	0	0	40
S4	SSH	0.000	2	0	1	60	0	0	60

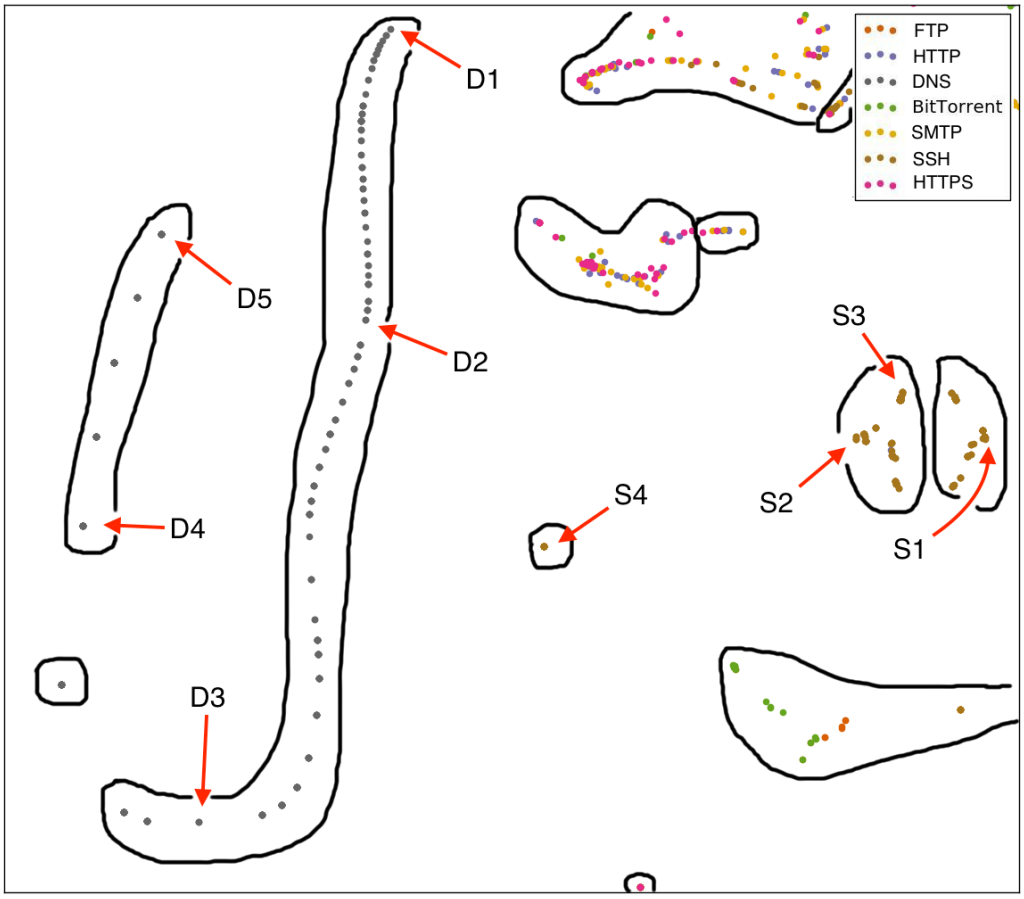

Fig. 4.4. Zoomed-in subsection of the *NetFlow* visualisation showing pure DNS and SSH clusters.

Although identified as belonging to three distinct clusters, points D1-D5 in Fig. 4.4 correspond to five flows that can be considered as belonging to one DNS chain. Examining the features observed for these flows in Table 4.3 indicates that the structure is entirely caused by a linear decrease in feature

values. Traversing the chain from D1 towards to D5, each flow has smaller values for the *bytes* and *Bpp* features than the last. Examination of the other DNS chains revealed similar situations, where the differences are only in the range of values in the *bytes* and *Bpp* features. In the remaining features, there is no variation in DNS flows in the entire dataset. This is logical considering that (1) all DNS flows in the dataset are UDP, and thus will never utilise the TCP *flags* feature, and (2) the DNS flows in the dataset are all single packet flows. Single packet flows have a *duration* of zero, thus statistics calculated with *duration* (*pps* and *bps*) are also always zero. A further consequence is that *Bpp* (bytes per packet) is always equivalent to *bytes*.

Flows S1-S3 in Figure 4.4 describe two nearby and pure SSH clusters, while S4 belongs to a more distant SSH cluster. From Table 4.4 we see that flows S1-S3 are, in fact, identical. Similar to the observed the DNS chains, these too are single packet flows. The difference here is that these SSH flows use the *flags* feature. The difference between S1-3 and S4 is purely in terms of *bytes* once again. S4 is observed to contain many stacked, identical flows, also sharing identical feature values. Finally, while these SSH clusters are pure, they account for a minor portion of the SSH flows. Referring back to the full visualisation in Fig. 4.3, it is clear that many SSH flows do not cluster as purely as these clusters.

Table 4.5 and Figure 4.5 focuses on second region of the *NetFlow* visualisation. The main structures we identify in this region are two nearby impure flow chains, and a large impure cluster.

Table 4.5: Table of NetFlow flows highlighted in Fig. 4.5.

Flow	Class	duration	flags	tos	packets	bytes	pps	bps	Bpp
C1	HTTPS	0.000	24	0	1	103	0	0	103
C2	FTP	0.000	24	0	1	101	0	0	101
C3	SMTP	0.000	24	0	1	86	0	0	86
C4	BitTorrent	0.000	24	0	1	79	0	0	79
C5	SSH	0.000	24	0	1	72	0	0	72
C6	HTTP	0.000	24	0	1	51	0	0	51
K1	SSH	2.752	24	0	3	308	1	895	102
K2	BitTorrent	9.088	24	0	10	873	1	768	87
K3	HTTP	29.874	24	0	2	106	0	28	53

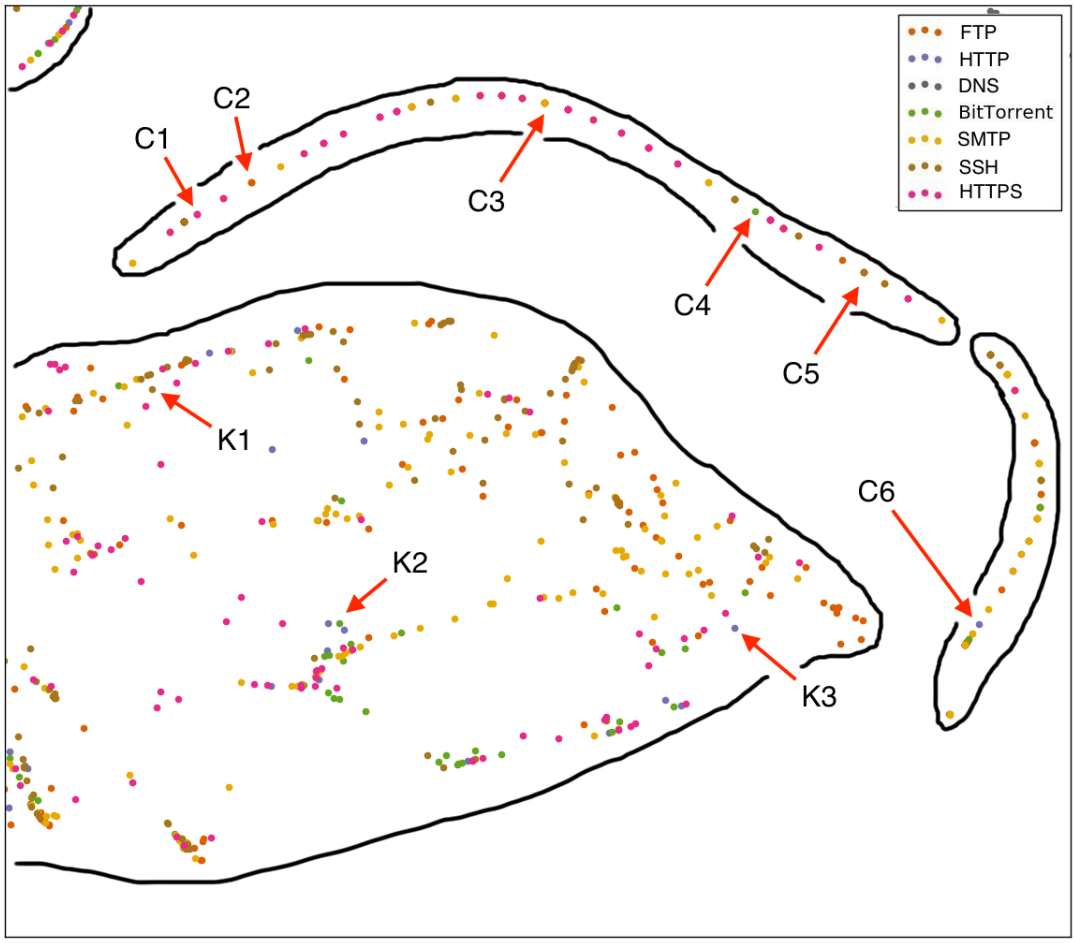

Fig. 4.5. A second subsection of the *NetFlow* visualisation, with flows highlighted from impure clusters.

Points C1-C6 identify chains that are reminiscent of the DNS chains examined earlier. Each flow in these chains is a single packet flow, and only varies in terms of *bytes* and *Bpp*. Unlike the DNS chains, these chains use the *flags* feature. However, all flows here share the same *flags* value. Despite the lack of variation across the features, each of C1-C6 originates from a different traffic class. That is, every non-DNS class appears in this single cluster. Flows K1-K3 belong to a more regularly shaped cluster. The increased spread observed here is attributed to multi-packet flows, which allow more variation in the features than in previous structures. The commonality of this cluster is that all flows are again equal in terms of the *flags* feature. Additionally, all non-DNS classes again co-occur in the cluster. Similar behaviour is observed in the remaining regularly-shaped clusters throughout the visualisation, where each cluster contains multi-packet flows with equal *flags*.

4.4.5 Visualisation of *bidirectional-NetFlow* Flows

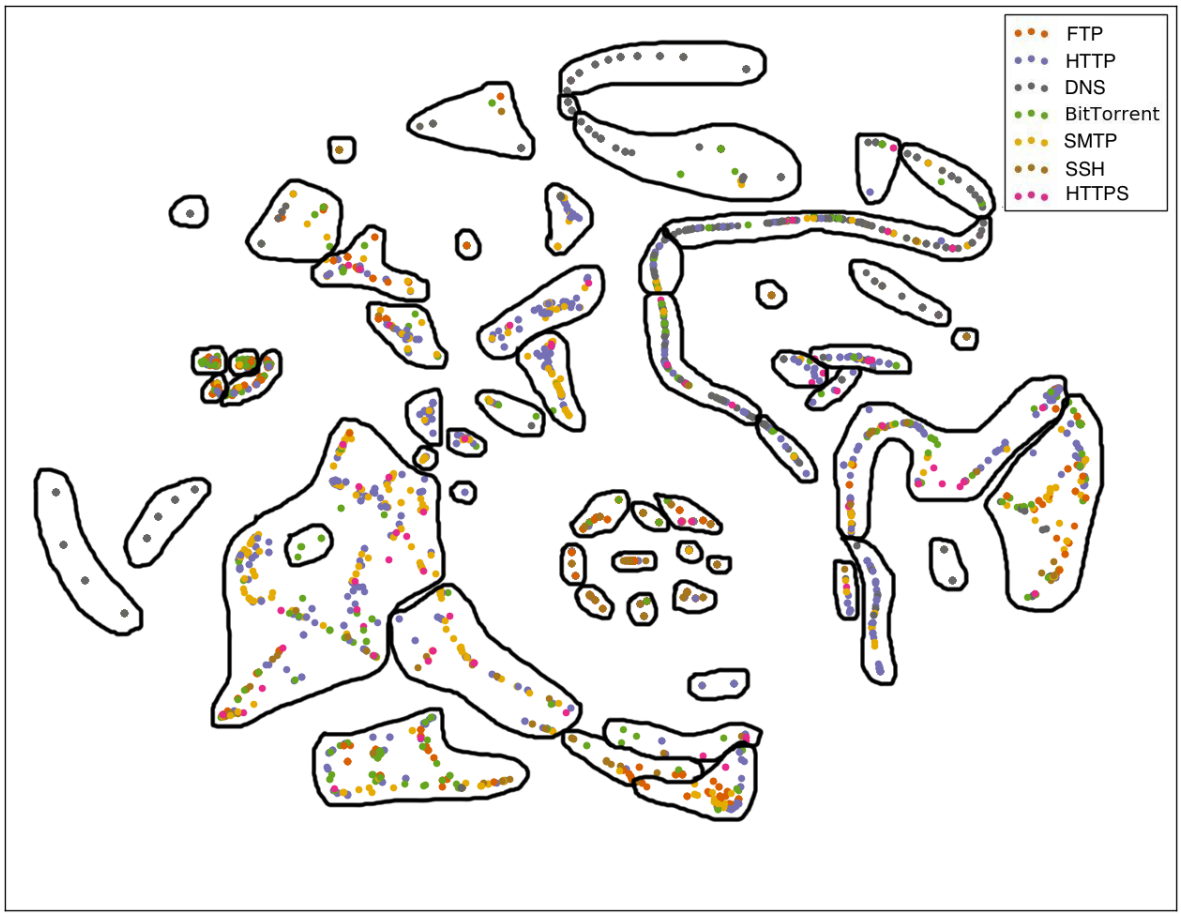

Fig. 4.6. Visualisation of the *bidirectional-NetFlow* dataset.

In the *bidirectional-NetFlow* visualisation (Figure 4.6), there are many similarities to the *NetFlow* visualisation despite the additional statistical features. From the visualisation, we identify the notable features as being chains of flows, large impure clusters, and small outlying clusters. The biggest flow chains (on the right hand side of the visual space) are exactly as was described by points C1-C6 in Table 4.5 in the *NetFlow* visualisation. Despite being bidirectional flows and despite the additional features, there were many flows still containing only single packets in one direction. Thus the chain shape is again caused by flows with linearly increasing values for the *bytes* feature. The critical difference to the *NetFlow* dataset is the classes present in these chains. In the *NetFlow* visualisation, chains were either DNS or non-DNS. Here, all traffic classes including DNS are present in the same chains. As the bidirectional feature set lacks the *flags* feature, the DNS class is more equivalent to the single-packet flows of TCP classes.

The larger clusters are again similar to those in the *NetFlow* visualisation. The more regular shape is caused by multi-packet flows with more spread in features. While the *NetFlow* clusters appeared to be

separated by *flags*, the lack of this feature causes more reliance on the remaining statistics. Finally, the outlying points interspersed throughout the visual space actually represent many flows with identical features, similar to what was observed in the *NetFlow* visualisations. For example, one point is actually 21 flows: 20 DNS flows and a single SMTP flow. A second point analysed is 27 DNS flows. Each stack contains identical flows in terms of the available features. This is primarily observed for DNS flows, indicating some separability based on the bidirectional statistics.

4.4.6 Visualisation of *wide flows*

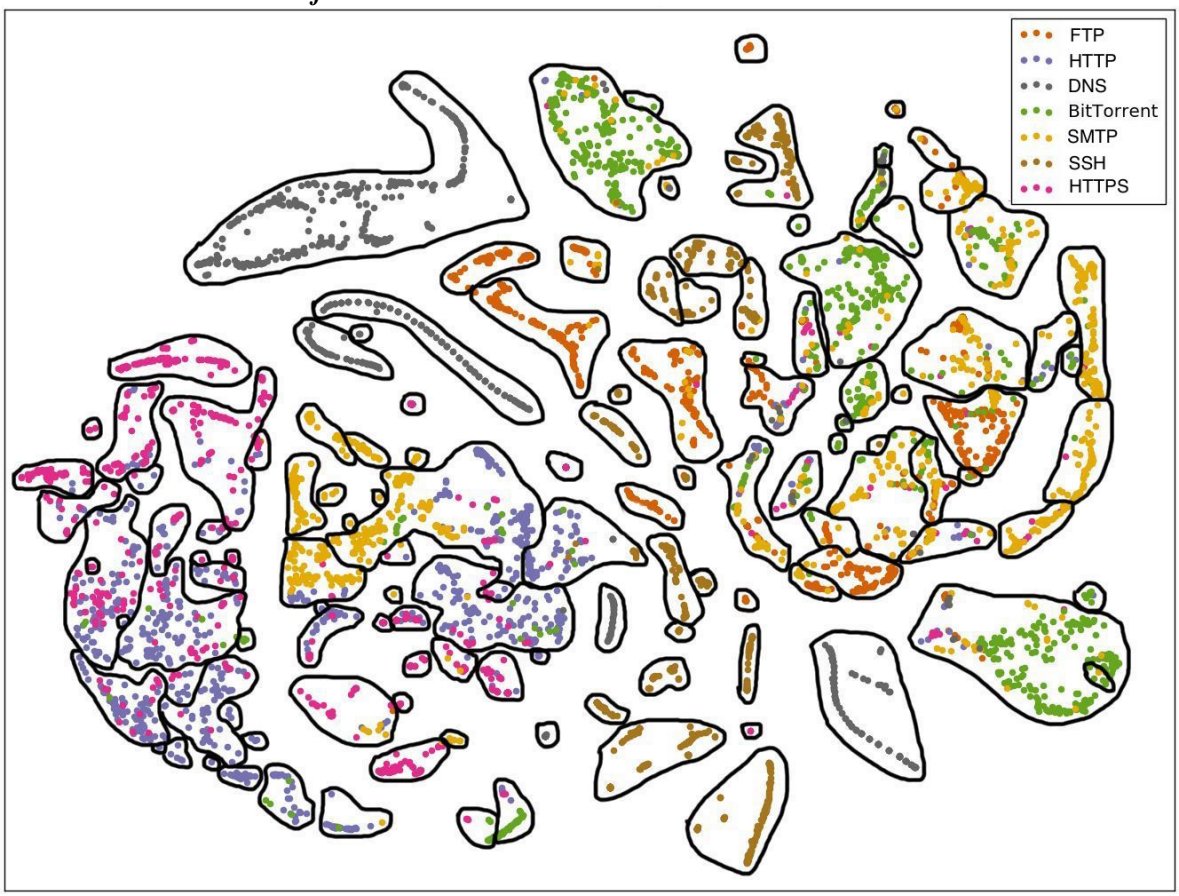

Fig. 4.7. Visualisation of the *wide* flow set.

In Figure 4.7, the visualisation of the *wide* flows is presented. As was observed in both NetFlow datasets, there are once again many more clusters than classes. This is most pronounced in the *wide* flow set, which has more and smaller clusters. Due to the larger number of features, there is significantly more variation and there are more contributing features to each cluster. Therefore, the underlying feature causing the clustering is less conspicuous than in the NetFlow datasets. It is immediately apparent that the *wide* clusters are more pure than those observed in the *NetFlow* and *bidirectional-NetFlow* visualisations. While some irregular shaped clusters are present, they appear to

be generally pure and the flows are well spread throughout the image. Furthermore, despite the large number of clusters, we observe a pattern of class distributions. Aside from outlying points, HTTP and HTTPS flows tend to be found towards the bottom-left of the visual space. DNS, SSH, and FTP tend to appear towards the centre. BitTorrent flows largely occur on the right side, and SMTP flows are generally found spread between the centre and right side of the image. In terms of statistical features, the observed pattern indicates more similarity and dissimilarity between certain pairs of classes. For example, HTTP and HTTPS frequently appear close together and thus are likely similar. HTTP flows rarely occur near FTP or SSH flows, indicating these classes are more dissimilar. This dissimilarity implies suitability for traffic classification.

DNS appears to again be a very pure class. First, we once again observe DNS chains. Due to the large number of features in this dataset, complete feature values for flows are not given here. However we observed that, despite the richer features set, these chains are again caused by single packet flows just as in the *NetFlow* data, with numerous flags and header features distinguishing these flows from other classes. Most notably, we also find a more cluster-like DNS structure towards the top of the visual space. Notably, this cluster contains more spread than the chains, caused by the bi-directionality of *wide* flows. These were typically flows with single packets in each direction, yet this allowed significantly more variation in the features than the unidirectional connections used by *NetFlow*.

While the *NetFlow* data hinted at the clustering potential of SSH flows in Figure 4.4, the class is clustered extremely well here. Next to DNS, the SSH flows appear to be the most pure clusters with only minor confusion with other classes observed. BitTorrent and SMTP generally belong to less pure clusters, with many of their clusters containing notable overlap with other classes. There is also significant overlap between HTTP and HTTPS flows. However, the large number of small clusters affords some relatively pure clusters for these classes.

4.5 Analysis and Discussion

The results throughout section 4.4 demonstrate an overall unsuitability of NetFlow formats for statistical network traffic classification. In 4.4.2, the supervised *wide* model supports earlier works regarding the sufficiency and strength of classifying traffic with purely statistical features. However, both the *NetFlow* and *bidirectional-NetFlow* datasets performed notably poorer.

The *NetFlow* model was better of the two, but the F-Measures indicate that the overall accuracy can largely be attributed to the DNS class. DNS was clearly easy to predict in the dataset, for reasons made clear by the feature importances and visualisations. The *NetFlow* feature importances indicated that the *flags* feature is overwhelmingly strong in this feature set, while the visualisation demonstrated that the

lack of the *flags* feature was the only distinguishing factor for DNS. Aside from this feature, variation was only seen in the number of bytes, which simply spread the DNS flows throughout the visual space. While the *flags* was a seen to be a strong predictor of the class, we consider it so strong simply because DNS is the only UDP class in our datasets. In the presences of other UDP classes, we expect the classification effectiveness for DNS flows to be notably poorer than observed here. Despite the low overall effectiveness found for this data, we therefore consider this an overestimation of the strength of the *NetFlow* feature set. This is confirmed when compared against the *bidirectional-NetFlow* visualisation. Despite the additional features, the *bidirectional-NetFlow* set performed more poorly overall and poorly on DNS due to the lack of *flags* feature. Despite the addition of bidirectional features, the visualisation in section 4.4.5 showed how the absence of the *flags* feature caused significant confusion between bidirectional DNS flows and other classes. When other UDP classes are present, we expect similar confusion among UDP classes in the *NetFlow* set. This exhibits a lack of generalizability in both NetFlow datasets. Considering the *wide* dataset, a more regular and pure cluster-like structure was observed for DNS. The variation and range of contributing features here were simply not present in the *NetFlow* and *bidirectional-NetFlow* dataset, demonstrating how this feature set can be generalized to other classes.

After DNS, the *NetFlow* feature set demonstrated potential at identifying SSH flows, with a relatively good F-Measure against the remaining classes. Figure 4.4 in section 4.4.3 demonstrated a tendency for SSH flows to separate into small, pure clusters. Again, given the feature values observed in these clusters, this can largely be attributed to the *flags* feature. Unlike DNS, multiple TCP classes are present and use this feature, indicating more reliable separability. However, these pure clusters accounted for only a small portion of SSH flows, with the visualisations showing regular confusion amongst other classes. Considering the remaining traffic classes, the chains and clusters observed exhibit the general lack of underlying structure and therefore discriminative ability in the *NetFlow* features. While the feature importances indicated that the *flags* and *Bpp* features were relatively strong features in the *NetFlow* feature set, but the visualisations demonstrated how these features were the *only* good discriminators. Due to the impure clusters and low classification effectiveness, we also consider these to be insufficient discriminators. The overall lack of strong features and lack of variation between classes confirms an insufficiency of statistical *NetFlow* features.

At first glance, bi-directionality appears to be a fundamental feature excluded from regular NetFlow flows and therefore the *NetFlow* features. Generally, bi-directionality is considered valuable for common flow analysis tasks [51], and the reasons are clear. Consider a client-server setting. In

unidirectional flows, one flow captures the client sending requests, and another flow captures the server responding to them. To classify traffic with statistical features, a unidirectional system therefore needs sufficient resolution to independently identify both request and response patterns of a traffic class. Meanwhile, a single, bi-directional flow that combines the network activity of both hosts will always more completely, and thus better, describe application patterns. This is demonstrated by the 98.35% classification accuracy obtained from the *wide* features. Furthermore, the Gini feature importances in 4.4.3 showed a good balance of highly ranked features in both directions, supporting the expected value of directional features. The importance scores were also relatively balanced, indicating a range of features contributing to effective classification, again indicating generalizability. However, similar value was not observed in the *bidirectional-NetFlow* feature set.

Despite the clear importance of directional features, the *bidirectional-NetFlow* feature set offered no clear improvement over the poor performance of the *NetFlow* features. In terms of the supervised classification effectiveness, it performed slightly worse than *NetFlow*. From the visualisation, we largely attribute the difference to the lack of *flags* feature in the bidirectional set. As was discussed earlier, this feature was found to be the strongest discriminator thus its absence made clear impact here. Despite the absence, the F-Measure for DNS was still notably high in *bidirectional-NetFlow*. The F-Measure was also marginally improved for the HTTP class. Together these indicate some good value in directionality. However, we still consider the *bidirectional-NetFlow* set to be a poor alternative to the *wide* set. As evidenced by chains and impure clusters in the visualisations, there is a fundamental lack of resolution in these features. Just as was observed in the *NetFlow* set, the features in the bidirectional set produce significant confusion due to lack of variation.

We attribute some low performance in the *bidirectional-NetFlow* set to difficulty in flow aggregation. In the NetFlow setting, unidirectional flows are aggregated using their 5-tuples. From this, there are clear and unavoidable sources for error. As flows are aggregated after they are completed, it is not guaranteed that bi-directional flows are correct. When two hosts have multiple complete connections within a short window, these flows can easily be confused during aggregation. The directionality of aggregated flows may also be reversed as a result. Furthermore, when a flow completes near the beginning or end of the capture window, its complimentary flow may be missed. In this case, these flows are the equivalent of unidirectional flows. When flows are captured across a small window of time (like was the case with our data), the effect will be more prevalent. While we found most of the flows aggregated, the chains of single, unidirectional flows in the *bidirectional-NetFlow* were evidence that this was not reliable. However, if the capture window used is too large, we expect flows may be

instead be aggregated too easily. The NetFlow processing tools used aggregates as many flows as meet the equivalence criteria. Thus a single bi-directional flow contains at least one unidirectional flow, but can contain upwards of three. We generally assume two flows per complete connection, so when flows are liberally aggregated, we expect that many of the available features would become even poorer indicators of traffic classes. Instead, they would better reflect the overall number of connections between hosts rather than underlying statistical patterns. Thus, while we consider bidirectional features important to traffic classification, their value is dependent on reliable capture.

Overall, when compared against the *wide* flow set, we identify the core reason for the poor effectiveness found for both the unidirectional and bidirectional NetFlow sets as follows. Considering the *wide* feature scores, statistics involving maximum values claimed the top three ranks. Naturally for the available classes, such features therefore appear most valuable. However, standard deviations, averages, and minimums were interspersed throughout the remaining top features, and the importances scores of these were well balanced. Compared to the *NetFlow* and *bidirectional-NetFlow* feature sets where only average statistics are available, this is a clear disparity. Considering the visualisations, the value of more features and more detailed statistics is clear. The *wide* set showed the largest number of clusters, and relatively very pure clusters. We consider the most significant feature of the image to be the large number of clusters per class. The large number is logical in the domain given the range of activity that can fall under a single traffic class. For example, retrieving a plain text document and streaming video can both be done with HTTP. The HTTP flows for these two activities would appear vastly differently in terms of statistical features. In the *wide* visualisation, the larger range of statistical features offers good separability, and overlap between classes is well resolved by having many small clusters. In the *NetFlow* sets, it is evident that the flows are largely just separated by their size.

Finally, the relative strength of the *wide* flows is furthered by evidence of generalizability. The range of underutilized features indicates how *wide* flow features can handle applications that were not explored here. For example, inter-packet time features were generally less important for our traffic classes. However, when time-dependent applications such as online video games and VoIP are present, it is expected that these temporal features are naturally stronger distinguishers. Similar arguments can be made for other weaker features. There is little evidence to suggest that NetFlow feature sets are flexible enough to better handle other kinds of applications.

4.6 Conclusions

Throughout this chapter, we analysed the potential for using typical NetFlow and NetFlow-like flow formats for statistical network traffic classification. Big data challenges and the prevalence of these

formats motivate the use of higher-level flow records, but existing machine learning approaches offered insufficient consideration for their applicability. Through a systematic evaluation, we compared the strengths of standard unidirectional NetFlow records and aggregated bidirectional records against the format typically used in existing machine learning approaches. Using supervised learning, we demonstrated a significant loss in classification effectiveness when using both NetFlow data sets. The poor performance of the NetFlow datasets was explained through an analysis of feature importances and visualisations. Through this analysis, it was demonstrated that the NetFlow feature set offered insufficient resolution to reliably distinguish classes. The bidirectional feature set hinted at the value of directional features, but the impreciseness of flow aggregation techniques resulted in equally poor performance. Simultaneously, the flow format from existing machine learning approaches confirmed its value, with a balanced range of features contributing to class distinction and demonstrating generalizability. For the applications in our datasets, maximum-value statistics and packet-oriented features were generally found to be the strongest features in the *wide* feature sets. These features were generally unavailable in both NetFlow feature sets. Having found significant value in the more detailed flow format, we therefore conclude that NetFlow records are insufficient for statistical traffic classification.

Chapter 5
A Novel Semi-Supervised Approach to Network Traffic Classification

In this chapter, we propose a semi-supervised machine learning approach to automatically identify network applications using purely statistical traffic flow properties. Our approach is based on a leading semi-supervised traffic classification approach [8], which supports flows generated by unknown applications. We propose a number of innovations to this method in order to significantly increase its classification effectiveness and efficiency. First, our approach introduces an alternate algorithm for mapping clusters to traffic classes. Second, we introduce a sampling method to significantly improve classification time. Finally, we introduce a methodology for feature selection with consideration for unknown classes.

5.1 Motivation

Semi-supervised approaches can currently be considered the most realistic machine learning framework for statistical flow classification. Using minimal pre-labelled data, existing approaches have shown good ability to counter the drawbacks of both supervised and unsupervised approaches [6], [8]. Namely, these approaches can both automatically identify known traffic classes and support unknown classes. However, while the effectiveness reported by previous approaches is good, certain traffic classes remained a challenge to identify. We aim to target these previously challenging classes. The open problem we address is, thus, how to improve the accuracy of traffic classification using statistical traffic flow properties. Furthermore, in situations where network managers are required to make timely decisions, efficient classification is crucial. Thus we further aim to minimize classification time.

5.2 Problem Statement

We are given a set of training flows $T = \{t_i \mid i = 1, 2, \ldots n\}$ and a set of testing flows $X = \{x_j \mid j = 1, 2, \ldots m\}$, generated on a single network. Each flow represents a bidirectional series of packets between two hosts, sharing the same source and destination addresses, port numbers, and protocol. Each flow has been generated by some known or unknown traffic class c. For each known class c, subsets of T exists such that $T_c = \{L_c \cap U_c\}$ and $\| L_c \| << \| U_c \|$, where L_c is the set of pre-labelled flows of class c and U_c is the set of unlabelled flows of class c. For any unknown class c, the subset of T containing flows of class c is $T_c = \{U_c\}$. That is, none of its flows are pre-labelled.

From T, we aim to create classifier $f(\mathbf{x}) = c$ such that when a flow $\mathbf{x}$ is given, a traffic class c is efficiently predicted. The traffic class c indicates that flow $\mathbf{x}$ was generated by a specific known class, or that it was generated by some unknown classes.

5.3 Our Proposed Approach

Figure 5.1 illustrates the details of our approach. The flow label propagation algorithm is first applied to a large training set containing a small number of labelled flows per class. The flow label propagation algorithm uses the correlated flows property of network traffic (described in 5.3.1) to automatically increase the number of labelled flows. Feature selection algorithms are then applied to this larger labelled set to identify the strongest feature subset. Next, clustering is performed on all training data, and then labelled flows are used to identify clusters as classes. Finally, the nearest cluster classifier predicts flow classes.

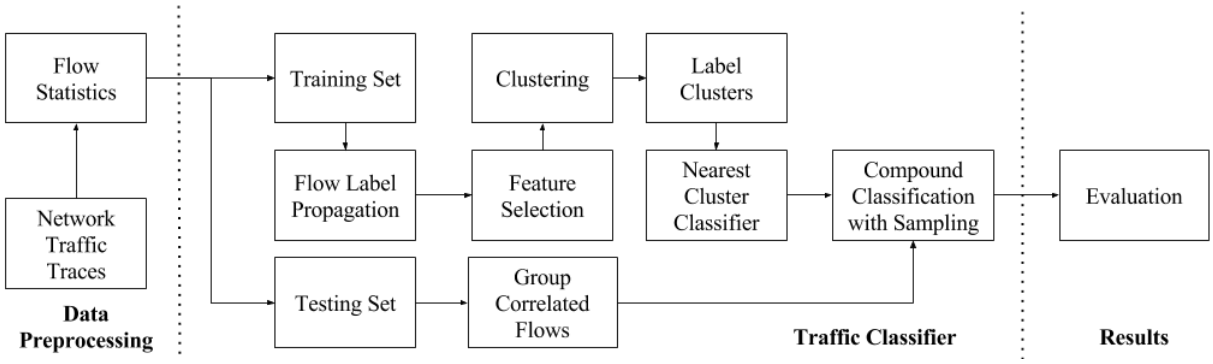

Fig. 5.1. System model for effective semi-supervised flow classification.

This system model is based on the works of Erman et al. [6] and Zhang *et al*. [8], with some key alterations. These approaches are described in more detail in section 2.3. For the remainder of this chapter, we will refer to these works as the Erman approach and the Zhang approach respectively. Like both of these approaches, the main advantage of our model is its ability to appropriately handle flows generated by unknown applications. Creating and identifying "unknown" clusters achieves this. However, we propose an alternative cluster-labelling algorithm (in section 5.3.3) for increased effectiveness. After flow label propagation, we then introduce feature selection to identify a stronger feature subset. We combine label propagation and unknown flow identification to allow feature selection algorithms to work effectively, as presented in 5.3.4. Finally, we introduce a sampling strategy into the classification stage, which can greatly improve the classification efficiency. This strategy is described in 5.3.5. Each major system component is described in more detail below.

5.3.1 Correlated Flows and Flow Label Propagation

Given a flow f_1 with some 5-tuple: source IP, source port, destination IP, destination port, and protocol, the correlated flow assumption is that any other flow f_2 sharing the same 3-tuple (destination IP, destination port, protocol) is correlated, i.e. generated by the same application. This assumption generally holds given that f_1 and f_2 are created within a limited window of time of one another. It has been shown that using this property can be an effective method of grouping flows into the same application class [4], [8], [52]. In the Zhang approach, this property is used to automatically extend a pre-labelled training set with *Flow Label Propagation*. Given a set of pre-labelled flows, *Flow Label Propagation* simply shares the label of any pre-labelled flows with any correlated unlabelled flows in the dataset. Due to the effectiveness demonstrated in the Zhang approach, we include the method in our model.

5.3.2 *k*-Means Clustering

We opt to use *k*-Means clustering as our clustering algorithm for its simplicity and sufficient scalability in terms of the number of clusters and training samples. The goal of *k*-Means clustering is to, through an iterative refinement procedure, find a way to separate *n* training examples into *k* distinct clusters such that inter-cluster variation is minimised and the variation between clusters is maximised. Algorithm 5.1 gives a high level description of the clustering algorithm.

```
Input:  training points X = {x_i | i = 1, 2, ... n}; number of clusters k
Output: k disjoint clusters

Init: Randomly initialise cluster centroids μ_1, μ_2, ... μ_k

1: Repeat until convergence:
2:    For i = 1 ← k:
3:          Update centroid μ_i
4:    For i = 1 ← n:
5:          Re-assign x_i to nearest centroid
```

Algorithm 5.1. *k*-Means Clustering

The main drawback of *k*-Means algorithm in the traffic classification context is that the number of clusters *k* must be chosen. Extensive work in the literature (such as [13] and [22]) has explored the effectiveness of *k*-Means under different settings of *k* in this domain. Due to extensive study of choosing this parameter, choosing *k* is not considered an issue here.

5.3.3 Cluster Labelling

The cluster-labelling algorithm introduced in this section is our proposed alternative to the algorithm used in the Erman and Zhang approaches. In these earlier works, clusters are mapped to applications by a simple majority vote, i.e. the label for some cluster c_i is the most common flow label observed in

c_i. If c_i has no labelled flows then it is an "unknown class" cluster. This is thus how unknown class support was achieved in these earlier approaches.

While we follow the same principle, we propose a novel cluster-labelling algorithm with two unique properties. First, we treat the unknown class as if it is known. Second, our algorithm allows clusters to be labelled as multiple traffic classes. For this reason we dub the algorithm *fuzzy cluster labelling* (FCL). The algorithm require a reasonable number of pre-labelled flows per class, which is achieved in our model by first applying label propagation (as described in 5.3.1) for the known classes.

```
Input: training flows T; set of k clusters trained on T;
flow label propagation function Propagate()
Output: traffic class labels, labels_i, for each cluster ci

1: for i = 1 ← k:
2:     # Unknown flow identification
3:      if c_i has no labelled flows:
4:             Identify x_i, the flow at the centroid of c_i
5:             Set label x_i = "unknown"
6:             Propagate(T, x_i)
7: for i = 1 ← k
8:      # Cluster labelling
9:      c_ij = number of flows labelled as class j in cluster ci
10:     labels_i = [argmaxj(c_ij)]
11:     max = cilabels_i[0]
12:     foreach traffic class j
13:            if j not in labelsi and c_ij * lax > max:
14:                   append j to labels_i
```

Algorithm 5.2. Fuzzy Cluster Labelling (FCL)

The majority vote approach in previous works ignores unlabelled flows, so a cluster with at least one labelled flow is always assigned a known class. Thus unknown classes are naturally down-weighted, regardless of their true prevalence. FCL instead aims to treat the unknown class equally with known classes. This means that the class is included during the cluster labelling process, rather than only accounting for leftover clusters. The challenge is that, due to the semi-supervised setting, it is expected that a significant portion of unlabelled flows belong to known classes. To account for this we automatically identify a core subset of unknown class flows. Firstly, clusters without any labelled flows are identified and the flows at their centroids are extracted. These flows are expected to be most representative of unknown classes and are labelled as "unknown". Then, to obtain a sufficiently large unknown class sample, flow label propagation is applied to labelled "unknown" flows. Cluster labelling then treats flows labelled as "unknown" flows just any other known class. Lines 1-6 in Algorithm 5.2 detail this process.

Lines 7-14 describe the *fuzzy* component of the algorithm. The *fuzzy* majority vote is achieved using the optional *lax* parameter. This parameter allows additional cluster labels in the case of no clear majority. A good size of *lax* is dependent on the purity of the clusters, but any $lax > 1$ allows non-majority classes to be cluster labels. The higher this parameter, the more freely additional class labels are assigned. We suggest that the choice of *lax* depends largely on the expected purity of clusters. The more impure clusters are expected to be, the more important we consider the allowance of multiple labels. With $lax = 1$, the algorithm is equivalent to majority vote with the addition of special treatment of unknown flows. Multiple labels are naturally resolved during classification (section 5.3.5). We apply compound classification that classifies all correlated test flows together via a majority vote of class labels. FCL simply allows each test flow to vote for multiple potential classes. The key intuition of FCL is thus to makes use of all labelled flow data available, while not ignoring the likely possibility of unknown classes.

5.3.4 Feature Selection

Irrelevant or unnecessary features can negatively impact the success of machine learning algorithms [53]. Feature selection methods thus aim to reduce a feature set to the most relevant subset. However, in our semi-supervised context, we initially have too few pre-labelled flows to expect effective feature selection. In our approach, we solve this problem is by first applying flow label propagation to the dataset. Once this is applied, a more substantial pool of labelled data for any given feature selection algorithms is available. However, applying feature selection only on this subset ignores the presence of unknown classes. If feature selection is applied purely to the labelled flows, the resultant feature set could be poor for any unknown classes present. Thus, to include feature selection in our semi-supervised context, we propose a preliminary clustering to discover unknown class flows. This is achieved similarly to unknown flow discovery in FCL, where label-free clusters are identified and their centroids are labelled as the unknown class. These centroids are then propagated to correlated flows. This preliminary clustering allows sufficient examples of both known and unknown class flows, on which feature selection algorithms can be applied. In this work, we reduce an extended set of statistical features by applying a decision tree approach to calculate feature importances. Features with relatively low importances are excluded. The decision tree algorithm, selected for its efficiency and simplicity, is described in section 4.3.3.

5.3.5 Compound Classification With Sampling

The correlated flow assumption allows us to apply a compound classification approach. The Zhang approach proposed classifying correlated flows together to reduce error. We instead propose using compound classification with sampling to improve classification efficiency. This approach is

complimentary to the proposed cluster labelling algorithms. With our consideration for the unknown class in our proposed cluster-labelling algorithm, we expect better labelling and thus fewer errors needing correction. With the assumption that all correlated flows belong to the same class, and the expectation that clusters are accurately labelled, it can be seen as unnecessary to classify every flow. We therefore propose a novel use of sampling to classify a minority subset of flows in each set of correlated flows. The majority class for each subset will therefore become the class for all flows in its bag.

5.4 Experimental Evaluation

Throughout this section, we thoroughly evaluate our proposed approach against the Erman and Zhang approaches. The Erman approach is the foundation of both the Zhang and our proposed approach and can be considered a baseline. The Zhang approach is a leading semi-supervised approach, shown to outperform other standard and state-of-the-art approaches when unknown applications are present.

5.4.1 Experimental Setup

Unless otherwise indicated, the settings for each experiment are as follows. For overall performance, we compare each approach on the *wide* and *waikato* datasets described in sections 3.1 and 3.2 respectively. We then focus on the *wide* dataset for deeper evaluation. The known classes for the *wide* experiments are HTTP, BitTorrent, SSH, FTP, and HTTPS. For the *waikato* dataset, the known classes are HTTP, SSH, HTTPS, and NTP. We simulate unknown classes with DNS and SMTP, using no pre-labelled flows for these classes in both datasets. We use 100 pre-labelled flows per known traffic class. This is shown to be an appropriate number of pre-labelled flows in [8]. For k-Means clustering, we k = 500 clusters for each experiment. The value of k chosen is shown to be suitable for obtaining pure clusters for network traffic [13]. Each experiment is performed with cross validation (section 5.4.2) and evaluated in terms of accuracy and F-Measures, as described in chapter 4.4.1. In our implementation of the Erman and Zhang approaches, we use the first 20 statistical features described in Table 3.2. These are the features used in the original approaches, which we will call the *original* feature set. The complete feature set in the same table is the *extended* feature set. In our approach, we use a compound classification sample of 10%. For FCL, we set lax = 2.5.

5.4.2 Cross Validation

When evaluating machine learning techniques, it is standard practice to evaluate a model on data unseen during its training to prevent overfitting. To ensure we both make use of all available data samples during both training and testing models, we employ m-fold cross validation. Thus we separate our dataset into m equal partitions, and each model is trained on $m - 1$ of these. The unused partition is used to evaluate performance of that model. This process is performed m times, each time excluding a

different partition for evaluation. The effectiveness and efficiencies reported using this technique are thus the averages of *m* different models. In these experiments, we have opted to use $m = 5$ folds.

5.4.3 Overall Classification Performance

Figure 5.2 presents the overall classification effectiveness on the *wide* dataset. In Figure 5.2(A), the accuracies of the Erman and Zhang approaches were observed to perform comparably at 77.65% and 75.88% accuracy respectively. Our approach classified with 94.17% accuracy on the same data. The F-Measures in Figure 5.2(B) shows that our approach performed as well or better than the comparable approaches for every traffic class. At 0.997, both our approach and the Zhang approach classified SSH equivalently and extremely well, only marginally outperforming the Erman approach's 0.995. With an F-Measure of 0.733 our approach again performed approximately equivalently to the Zhang approach's 0.731 on the HTTPS class. Both were clear improvements over Erman's 0.526. All three approaches were also relatively good at classifying the HTTP class, but our approach clearly led with an F-Measure of 0.954, an improvement of approximately 0.08 over the Zhang approach and 0.25 over the Erman approach.

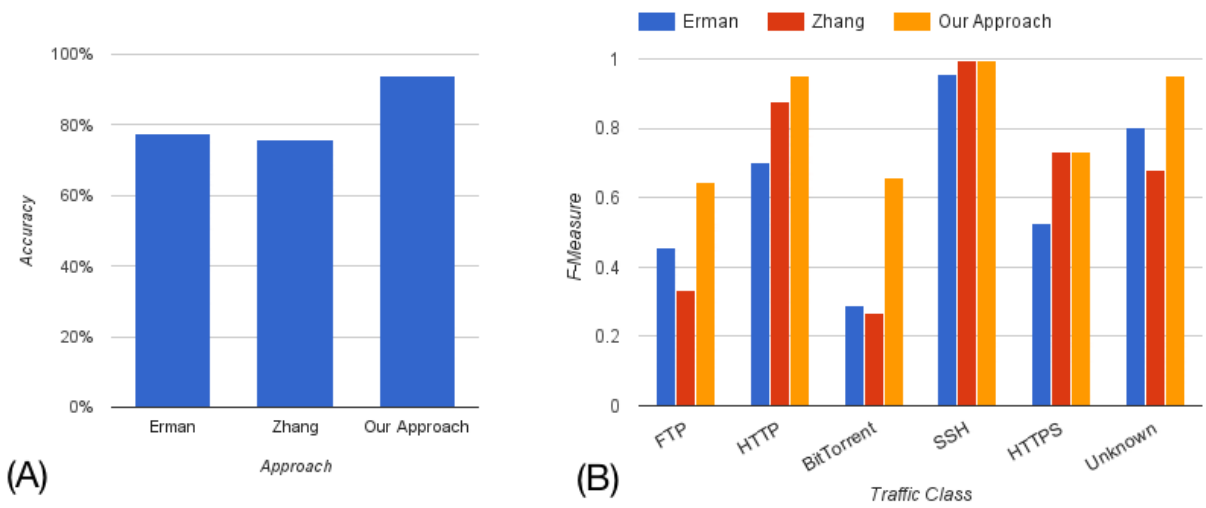

Fig. 5.2. Overall classification accuracy (A) and per class F-Measure performance (B) of our approach against the Erman and Zhang approaches on the *wide* dataset.

The most notable improvement was observed in the BitTorrent class. At 0.289 and 0.266, both the Erman and Zhang approaches indicated poor ability to identify this class. Our approach classified the same class with an F-Measure of 0.659, a significant increase of approximately 0.4. Large improvement can also be seen on the FTP class. Our approach classified FTP with a score of 0.646, compared to Erman's 0.458 and Zhang's 0.332. Lastly, the unknown class was best classified by our approach with an F-Measure of 0.952. At 0.805, the Erman approach still performed notably better than the Zhang approach's 0.681.

The overall accuracy of each approach on the *waikato* dataset is shown in Figure 5.3(A). On this dataset, the performance gap between the Zhang and Erman approaches was more pronounced. At 79.42% accuracy, the Zhang approach this time outperformed the Erman approach by a significant 10.75%. Our approach again had the highest accuracy at 89.90%, clearly above both other approaches.

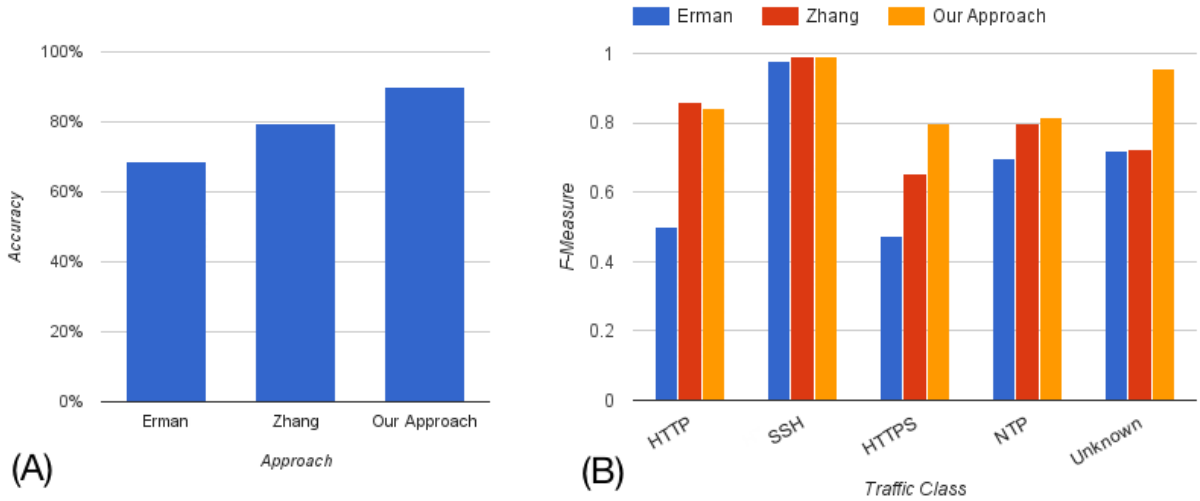

Fig. 5.3. Overall classification accuracy (A) and per class F-Measure performance (B) of our approach against the Erman and Zhang approaches on the *waikato* dataset.

The F-Measures in Figure 5.3(B) are similar to those described in the previous dataset. Again all three approaches performed very well for SSH, with the lowest being Erman at 0.979. The F-Measures for the Zhang approach and our approach were 0.991 and 0.992 respectively. Our approach was slightly outperformed on the HTTP class. On this class, the Zhang approach reported an F-Measure of 0.860, compared to 0.843 for our approach. Both approaches were notably better than the 0.498 from the Erman approach. Our approach achieved the highest results on the remaining classes. On the unknown class, the Zhang and Erman approaches performed approximately equally with F-Measures of 0.724 and 0.721. At 0.957, our approach performed best by a significant margin of over 0.23. Similarly on HTTPS, our approach led by 0.147 over the Zhang approach and 0.325 over the Erman approach. Finally, our approach also performed best on the NTP class at 0.818, followed by Zhang at 0.796, and Erman at 0.699.

Additionally, our approach used 10% classification sampling to achieve these results. On the *wide* dataset, our approach classified approximately 100,000 flows in an average of 46.26 seconds, less than a quarter of the 207.01 seconds required for the alternative approaches. Our approach also classified approximately 120,000 flows in the *waikato* dataset in 43.31 seconds compared to 259.39 seconds for the Erman and Zhang approaches, thus classifying in approximately 17% of the time.

5.4.4 Effectiveness Under Different Unknown Classes

In this section, we further evaluate the overall effectiveness of each approach when varying the simulated unknown classes. The *wide* dataset is used for each of these experiments.

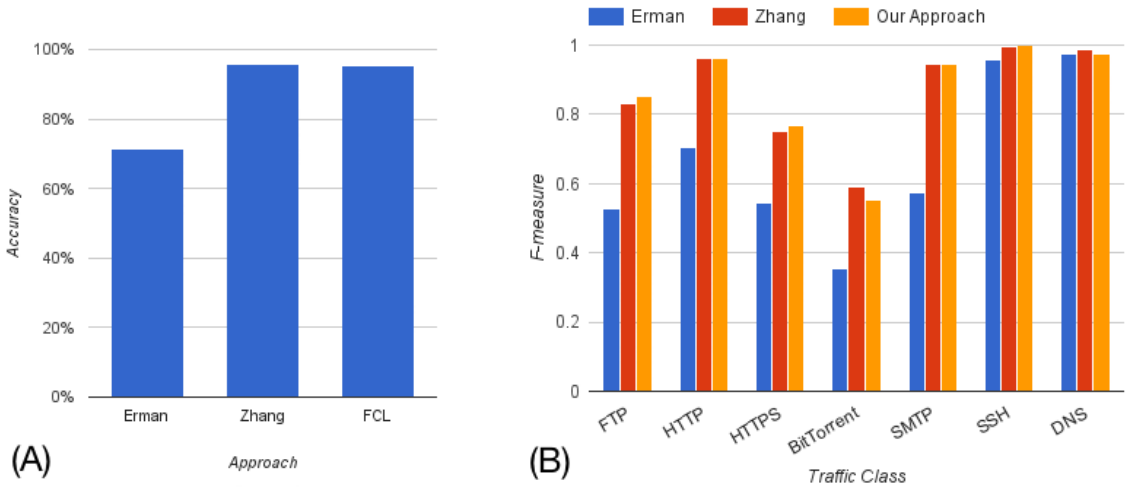

Fig. 5.4. Overall accuracy (A) and per class F-Measure performance (B) when no unknown classes are present.

In Figure 5.4, the results for zero unknown classes are presented. In this setting, the Zhang approach and our approach performed practically equivalently at 95.84% and 95.33% accuracy (Figure 5.4(A)). The Erman approach performed considerably poorer at 71.60% accuracy. The near-equivalent accuracies of our approach and the Zhang approach are explained by the F-Measures in Figure 5.4(B), where both approaches are comparable for every traffic class. The biggest difference between our approach and the Zhang approach was on the BitTorrent class, where the Zhang approach scored a marginal 0.038 higher. Our approach tended to marginally outscore the Zhang approach on the remaining classes. Meanwhile, the Erman approach performed worst on all classes. This was less evident on SSH and DNS flows, but the Erman approach performed between 0.20 and 0.37 worse than our approach and the Zhang approach on the remaining classes.

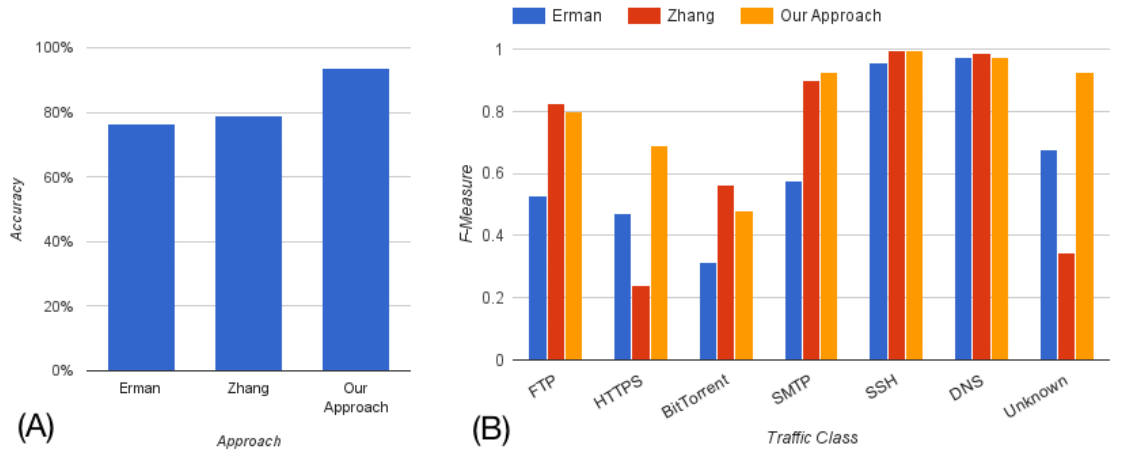

Fig. 5.5. Overall accuracy (A) and per class F-Measure performance (B) with HTTP as the unknown class.

With HTTP as the unknown class, our approach led with the overall best accuracy by approximately 15% (Figure 5.5(A)). The Erman and Zhang approaches performed similarly at 76.67% and 79.12% accuracy, while our approach reported 93.79%. All approaches again perform comparably on SSH and DNS (Figure 5.5(B)). The Zhang approach slightly outperforms our approach on FTP (a difference of 0.028) and BitTorrent (a difference of 0.083). However, our approach makes up for this by much more significant improvements on HTTPS and the unknown class. On HTTPS, our approach resulted in an F-Measure of 0.688; for the Zhang approach, the F-Measure was just 0.240. The Erman approach had midrange performance at 0.471, but was still significantly lower than our approach. For the unknown class, the Erman approach again outperformed the Zhang approach at 0.678 against 0.346. At 0.924, our approach was the clear leader for the unknown class.

5.4.5 Effect of Feature Selection

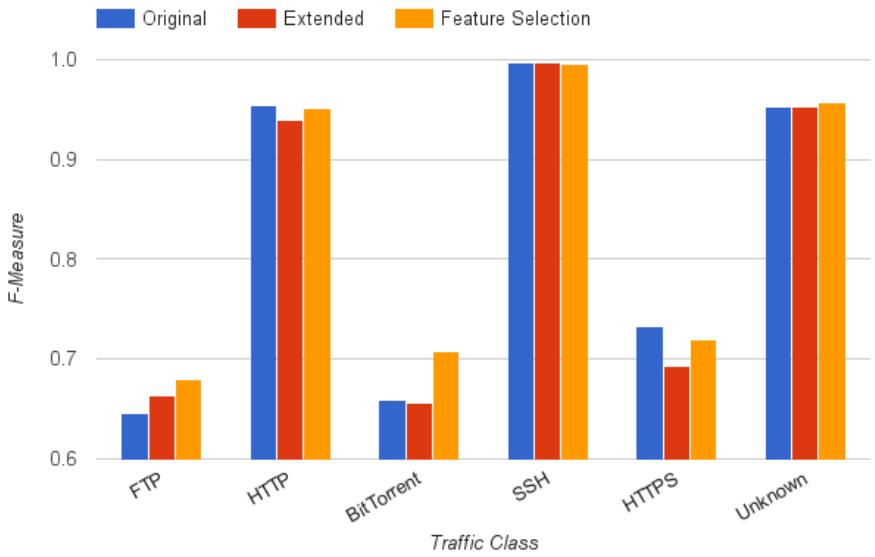

Fig. 5.6. Impact of feature selection against the original and extended feature sets.

The "feature selection" feature set reduced the extended feature set from 39 features to the 25, the impact of which is shown in Figure 5.6. All three feature sets performed near-equivalently on the three strongest classes (SSH, HTTP, and unknown). The extended feature set dipped slightly for HTTP at 0.940, compared to 0.954 and 0.951 for the original and reduced feature sets. More substantial difference was observed for the weaker classes. The original feature set performed best on HTTPS at 0.733, followed by the feature selection set at 0.719. The extended set was notably weaker at 0.693. The feature selection set performed best on both FTP and BitTorrent. On BitTorrent, the feature selection set resulted in an F-Measure of 0.707. For the original and extended feature sets, the result was 0.659 and 0.657. On FTP, the original feature set was the poorest performing set with an F-Measure of 0.646. This was marginally improved by the extended feature set, scoring 0.663. The best

performance for this class was again by the feature selection set, with 0.680. The key result here is that the feature selection set performed as well or better than the extended set on every class.

5.4.6 Effect of varying *lax* in FLC

The per-class classification effectiveness under different setting of *lax* is shown in Figure 5.7. We consider a low setting of *lax* = 1, an normal setting of *lax* = 2.5, and two very large settings of *lax* = 5 and *lax* = 10. The normal *lax* setting resulted in consistently high F-Measures. While only small increases, the increases were consistent across multiple classes. The largest increases are observed in the weakest classes. In HTTPS, the increase was 0.031 against the low *lax* setting. For BitTorrent and FTP, the increases were approximately 0.015 and 0.014 respectively. While the increases are small, the good choice of *lax* was consistently equal to or better than the low *lax*.

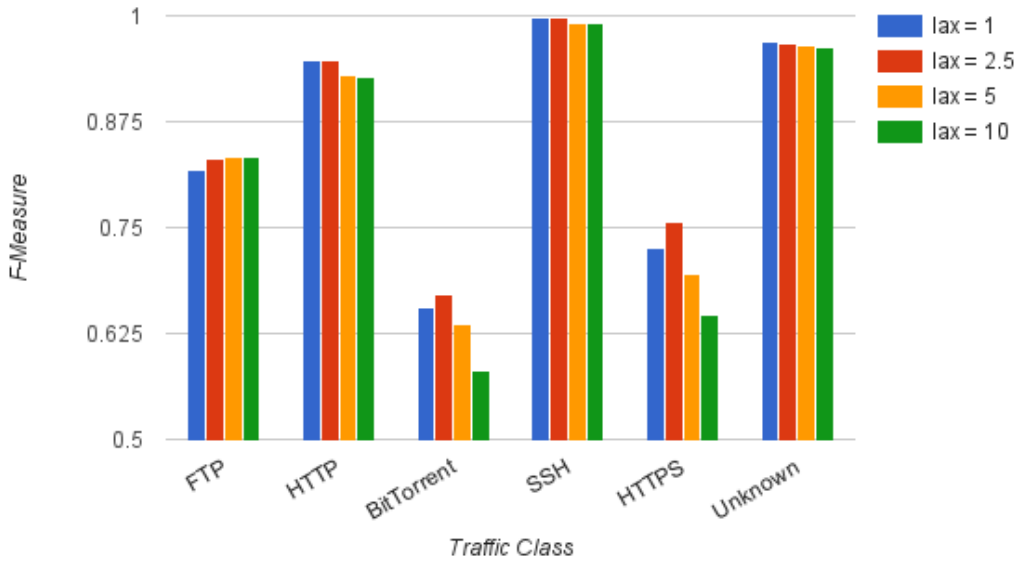

Fig. 5.7. Impact of varying *lax* on F-Measures.

Considering the two large *lax* settings, classes that were already predicted well were seen to be predicted quite robustly. The decreases in F-Measures for the SSH and unknown classes were largely insignificant. All values of *lax* > 1 were seen to marginally improve FTP. The large *lax* settings were more detrimental for BitTorrent and HTTPS. *lax* = 5 performed worse than the low and regular *lax* on both classes, and *lax* = 10 performed worse again.

5.4.7 Classification Efficiency

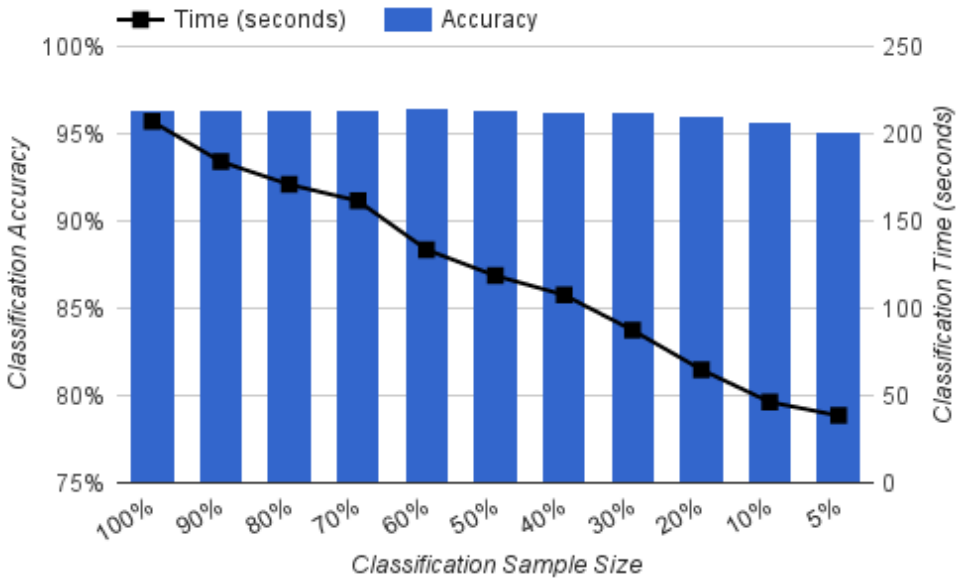

Fig. 5.8. Impact of classification sample size on overall accuracy and classification time.

In this section, we evaluate the impact of applying compound classification with sampling to the overall classification efficiency and effectiveness. Each classification time reported in Figure 5.8 is the average of 10 runs on a 2012 MacBook Air with a 1.8 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 memory. We report the accuracies and classification times for sample sizes varying from 100% to 5%. The 100% sample is equivalent to the compound classification employed in the Zhang approach. Overall, the accuracy is seen to be quite resilient to variations in the sample size. However, the classification time significantly reduces when sampling is employed. At 100% sampling, the accuracy averaged 96.39% and took in 207.01 seconds. At a 90% sample, the classification time reduces to 184.05 seconds with no change in the accuracy. Continuing this trend, the overall accuracy varies by just 0.23% from classification samples of 80% to 30%, with a minimum of 96.29%. Despite near-equivalent accuracies, the smaller sample sizes naturally further decreased classification time. From 20% to 5%, more loss in accuracy is observed. The 20% sample size resulted in 96.02% accuracy with a classification time of 64.86 seconds. At 95.70% accuracy, the 10% sample required 46.22 seconds. Finally, the 5% sample achieved 95.06% accuracy in just 38.52 seconds. This final sample had the most significant decrease in accuracy at 1.33% less than the 100% sample. However, it completed its classification in less than 19% of the time.

5.5 Analysis and Discussion

Our approach was demonstrated to achieve consistently and significantly improved classification effectiveness and efficiency over both the Erman and Zhang approach. In terms of accuracy, we found

an improvement of 18.29% over the Zhang approach and 23.73% over the Erman approach. Furthermore, our approach achieved the highest accuracy by a significant margin for all evaluated settings of the unknown class. In terms of individual classes, our approach matched or improved the F-Measures of every class compared to the alternative approaches in most settings. When our approach performed poorer on individual classes, it was only by minimal margins. Our approach also achieved this effective classification in under 20% of the time.

We largely attribute the significant effectiveness improvement to our unique handling of unlabelled flows. In the Erman approach, an equal number of pre-labelled flows are used per known class, regardless of the overall class size. The Zhang approach begins with the same set of pre-labelled flows, but applies flow label propagation to automatically increase the number of labelled flows. While it is generally expected that having more labelled flows would result in better effectiveness, we instead observe a strong bias towards known classes that can be very detrimental to classification effectiveness. The bias stems from flow label propagation working *too* effectively for some classes, and thus we found that the Zhang approach risks significantly under-weighting the unknown class. The opposite can be said for the Erman approach, which instead risks over-weighting this class.

To illustrate this point, we consider the original settings in 5.4.3, where SMTP is an unknown class. First, we acknowledge that our implementation of the Zhang method found notably poorer effectiveness for some classes in this setting [8]. We attribute the discrepancy to two factors. First, although we both use similar *wide* datasets from the same network, we use different samples of data. Second, and more crucially, we include an extra traffic class (FTP) that was not used in [8]. This class is observed to lower the performance of the Zhang approach, which we attribute to its extremely successful label propagation. That is, flow label propagation was observed to share 100 randomly pre-labelled FTP flows to up to 99% of unlabelled FTP flows. However, we also found that SMTP and FTP flows frequently occurred in the same clusters. Because FTP flows were so effectively labelled, and because no SMTP flows were labelled, almost every unknown (SMTP) cluster was incorrectly labelled as FTP. The result is that when SMTP was an unknown class, near perfect recall but very poor precision resulted in an F-Measure of just 0.332 for FTP. For the unknown class, the opposite was observed. The F-Measure for the unknown class was higher at 0.681, but this is largely attributed to the DNS flows also being unknown in this setting. When DNS was treated as a known class, we observed that the class is particularly easy to identify. This F-Measure for the unknown class was therefore found to be considerably lower than both our approach and the Erman approach. A similar situation was observed for the BitTorrent class. When we varied the unknown class to HTTP, the

Zhang method performed significantly better on FTP, but performed worse on HTTPS. This was again caused by high recall and low precision. When no classes were unknown, the Zhang approach performed extremely well on all classes. Generally, when unknown classes are present, the Zhang approach exhibits very high precision but poor recall. This demonstrates how the Zhang approach can significantly and detrimentally over-weight labelled flows and simultaneously under-weight the unknown class. More intuitively, the opposite weakness was seen in the Erman approach. As was acknowledged in the original work [6], a small number of pre-labelled flows can leave many clusters devoid of any labels, and thus the unknown class is easily and frequently over-represented. Our implementation confirmed this weakness. Notably, under some settings, the more precise use of labelled flows allowed the simpler Erman approach to outperform the Zhang approach.

Our approach thus demonstrates the importance and value of carefully treating the unknown class in a semi-supervised setting. While the Erman approach over-represents the unknown class, and the Zhang approach under-represents it, we found consistent improvements due to a more balanced representation through FCL. The use of centroids to extract unknown flows provides for the most confident estimation of an unknown class flow. Due to flow label propagation, using label-free clusters was found to be sufficiently reliable in our experiments. Applying label propagation to our labelled unknown flows produced a sufficiently large unknown class set that could then be treated equally with any other class. Due to this balanced treatment, our approach was able achieve much more stable effectiveness regardless of the unknown class settings. However, due to our proactive handling of unknown flows, the setting with no unknown classes is critical for considering our approach effective. This setting was found to be most ideal for the Zhang approach, which significantly outperformed the Erman approach simply due to having more labelled flows. In our approach, there are very few label-free clusters, due to label propagation, so our special treatment of the unknown class is not observed to be detrimental. In the no-unknown setting, our approach was very comparable to the Zhang setting at just 0.51% lower accuracy. Thus, even in the best-case setting for the Zhang approach, our approach also performs extremely well by comparison.

The *lax* parameter in FLC represents our second source of balanced class treatment. For our traffic classes, varying the *lax* made only marginal difference. At *lax* = 2.5, we did, however, find improved F-Measures for the FTP, BitTorrent, and HTTPS classes. These are the three smallest and weakest classes in our dataset, yielding less pure clusters. Thus the allowance of additional labels was seen to offer some small but notable value here, rather than being overwhelmed by the dominant classes as in majority vote. The results were relatively unaffected by increasing *lax* to very high values for SSH,

HTTP, and the unknown class. We expect that *lax* would have more substantial impact when more impure classes are present. Most importantly, a more conservative choice of *lax* (e.g. between 1 and 2.5) is rarely expected to worsen classification effectiveness, as additional labels are only ever included if they have sufficient presence in a cluster. Therefore, while it is a parameter that must be chosen, a conservative choice of *lax* has no obvious downside. Additionally, our proposed clustering labelling algorithms are seen as efficient in terms of computational complexity. Let n represent the number of flows in a cluster and c represent the number of traffic classes. After unknown flow identification, the total time complexity for our labelling algorithm is thus $O(n + c)$. As $c << n$, this is approximately equivalent to the $O(n)$ of the majority labelling method from [2], [4].

The feature selection component of the system was shown to be able to perform as well or better than the extended feature set on every traffic class. This demonstrates the validity and value of using the intermediary model to extract unknown class flows. Despite the cost of an intermediary model, we consider this step to be crucial in applying feature selection when unknown classes are expected to present. Neglecting the unknown flow extraction will result in a feature set that is likely strong for the known classes, but unpredictable on the unknown class. Furthermore, we found notable overlap between the reduced feature selection set and the original feature set, indicating that the features used in the Zhang and Erman approaches are sufficient for the traffic classes explored here. However, the small yet notable improvements on the weaker traffic classes further indicate the value of this component. Despite the training cost, we consider combining an extended feature set with our feature selection approach to be generalizable to other traffic classes.

Finally, we consider the value of introducing sampling into classification. The overall accuracy was observed to be extremely robust under different sample sizes, with practically no loss in accuracy between 100% and 30% samples. Even at a 5% sample, the difference in accuracy against the 100% sample was a just 1.33%. The decrease in accuracy naturally comes from assigning sample flows as the label of impure clusters. While in theory it may seem that classifying large groups of flows by a minority could therefore be extremely detrimental, in practice the error is small due to largely accurate labelling. In situations where efficient results are ideal, there is clear value in using sampling-based compound classification. Even when timely classification is not required, more conservative bag samples of approximately 30% can classify in well under half the time with no impact on effectiveness. Additionally, this classification method is not restricted to our approach. Rather, this approach can be applied to improve the classification efficiency of *any* traffic flow classifier.

5.6 Conclusion

This chapter presented a new approach to semi-supervised network traffic classification. An overall accuracy upwards of 94% demonstrated the effectiveness of classifying traffic with our approach. Furthermore, consistent improvements in per-class F-Measures demonstrated the effectiveness of our approach at targeting previously difficult-to-classify classes. Through an empirical evaluation on standard benchmark datasets, our approach was demonstrated to consistently and significantly outperform the state-of-the-art method on which it is based. We further introduced a safe strategy for introducing feature selection with careful consideration for the presence of unknown classes to further improve effectiveness. Finally, we demonstrated how classification time could be considerably reduced at little-to-no cost to effectiveness.

Chapter 6
Towards Realistic Unsupervised Flow Classification

In this chapter, we combine set theory and cluster aggregation with the goal of addressing the fundamental drawback of existing unsupervised network flow models. We propose an original, preliminary system that demonstrates strong potential for more realistic and interpretable unsupervised flow models. Our approach combines a novel correlation property with a flow-specific cluster similarity function and aggregation algorithm. By reducing both the number of clusters and sets of flows, we demonstrate how the complexity of unsupervised models can be significantly reduced.

6.1 Motivation

Given that network traffic is considered Big Data and the difficulty of obtaining labelled data, clustering approaches are especially appealing for flow classification and other related network problems. However, existing clustering approaches require many more clusters than classes [3], [13], [16]. Throughout our work in Chapter 5, we demonstrated how using 500 clusters for 6 to 7 traffic classes resulted in sufficiently pure clusters for an effective semi-supervised classifier. In unsupervised settings, such a large disparity between number of classes and clusters causes extremely poor model interpretability. This remains a fundamental weakness of these approaches. Current unsupervised approaches require laborious manual analysis or approximation methods to identify clusters. For example, a common approach is to analyse a sample of flows from each cluster. The difficulty and imprecision of current unsupervised model analysis methods motivates research towards improving their interpretability.

To the best of our knowledge, the only existing work addressing this problem aggregates clusters using packet payload analysis [16]. However, not only did the approach require external knowledge about the true classes, but the need to capture and use full packet payloads is a hefty requirement on any large network in the face of Big Data challenges. Furthermore, we expect the approach would fail when traffic is encrypted. This previous approach is described in more detail in section 2.1. For a more generalizable and realistic approach, we therefore focus on improving model interpretability purely using statistical flow features and correlated flow properties.

6.2 Problem Statement

We are given a set of traffic flow clusters $K = \{k_i \mid i = 1, 2, \ldots n\}$ that was created on a set of unlabelled flows. For each flow f, only its 5-tuple (source IP, source port, destination IP, destination port, and protocol), and exclusive membership to some cluster k is known. Each flow was generated by some

unknown traffic class c out of a set of C traffic classes. Furthermore, the true number of classes $| C |$ is unknown, and $n >> | C |$. If the true class for a cluster k is c_k where c_k is most common true class present in k, we aim aggregate pairs of clusters k_1 and k_2 if and only if $c_{k1} = c_{k2}$. The resultant cluster set $S = \{s_j \mid j = 1, 2, \ldots m\}$, where $m < n$. As m approaches $| C |$, model interpretability improves.

6.3 Approach

In our approach, we use the flow correlation properties described in 5.3.1 to generate a "bag-of-flows" model, where each "bag" is a set of flows originating from the same traffic class. Using this model, there are three main components in our model for unsupervised traffic flow model simplification (Figure 6.1). First, we introduce Cross-Cluster Correlation in section 6.3.1. This is a novel technique for reducing the number of "bags" of correlated flows. This technique is first used to increase cluster affinity information for more effective aggregation, and then later used again to simplify analysis. The second component we introduce is a function for quantifying cluster similarity in section 6.3.2. Our final component in section 6.3.3 uses this similarity function in a cluster aggregation. Each component is described in more detail throughout this section.

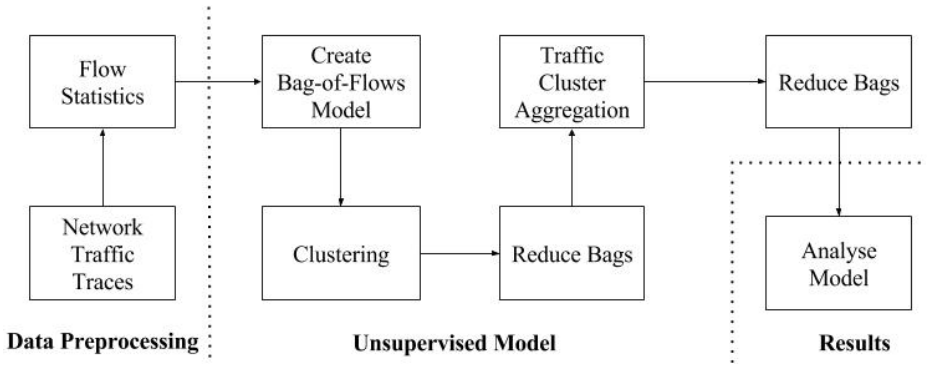

Fig. 6.1. System model for generating simplified unsupervised traffic classification models.

6.3.1 Cross-Cluster Correlation

In a bag-of-flows model, each bag contains at least one flow and appears in at least one traffic cluster. Given that all flows in any given bag are expected to belong to the same traffic class, and given that bags can appear in multiple clusters, we propose Cross-Cluster Correlation (CCC) to determine equivalent pairs of bags. Equivalent bags are merged to significantly reduce the number of bags, and to increase the value of each bag.

Consider two bags A and B. With reasonably pure clusters, we consider A and B equivalent if they always co-occur. That is, if C_A is the set of clusters in which bag A is observed, and C_B is the set of clusters in which bag B is observed,

$$A = B \; if \; \frac{| C_A \cap C_B |}{| C_A \cup C_B |} = 1 \tag{14}$$

For more aggressive bag reduction, we loosen this rule to define CCC in Equation 15.

$$\text{Cross-Cluster Correlation:} \quad A = B \; if \; \frac{|C_A \cap C_B|}{|C_A \cup C_B|} \geq p \tag{15}$$

That is, two bags are equivalent if they reach a minimum overlap threshold p. Given that traffic clusters are generally expected to be pure, we propose that frequent co-occurrence is a good measure of equivalence. From the looser bag reduction, we apply basic logical equivalence to further aggregation. That is, for three bags A, B, and C,

$$\text{If } A = B \quad \& \quad B = C \; \therefore \; A = C \tag{16}$$

Therefore, just as the bag-of-flows model reduces the number of variables via flow aggregation, Cross-Cluster Correlation exploits cluster memberships for further reduction.

Merged sets of bags are inherently valuable. For general uses, bigger bags contain more equivalent flows, and thus each carries more information about the network. Additionally, analysing a single flow from a bag identifies all flows in the bag. Therefore, merging bags can simplify model analysis. In our system, we primarily use CCC as a pre-processing step for cluster aggregation. By bag reduction through equivalence, we aim to increase affinity between clusters belonging to the same traffic class. This is an important component of our cluster aggregation algorithm, which we describe in 6.3.3.

6.3.2 Flow Cluster Similarity

Cluster aggregation requires a metric for evaluating the similarity of any given pair of clusters. General-purpose aggregation techniques use distance-based criteria to determine cluster similarity, where nearby clusters are merged. In the traffic flow cluster setting, different clusters represent different statistical patterns of a small pool of classes. Since classes can exhibit a wide range of statistical patterns, distance between clusters does not reliably indicate the equivalent classes. Therefore, we define a domain-specific similarity function for evaluating the class similarity of cluster pairs. Equation 17 describes the function, where X_B is the set of bags in cluster X, $|X|$ is the total number of flows in X, and $|X_b|$ is the number of flows in X belonging to a single bag b.

$$Similarity(X,Y) = \left(1 - \left(\sum_b^{X_B \cap Y_B} \left| \frac{|X_b|}{|X|} - \frac{|Y_b|}{|Y|} \right| \times \frac{1}{2}\right)\right) \times \left(\left(\frac{\sum_b^{X_B} |X_b|}{|X|} + \frac{\sum_b^{Y_B} |Y_b|}{|Y|}\right) \times \frac{1}{2}\right) \tag{17}$$

The left half of Equation 17 indicates cluster similarity via the bags present in both clusters. More specifically, the absolute difference in proportions of each shared bag in each cluster is summed. To compensate for any cluster impurity, this sum is used to identify if shared bags are important to both clusters. If a bag is proportionally important to both or neither cluster, the difference is small. If the

bag is important to only one cluster, this could indicate an impurity and thus the discrepancy is highlighted. The difference is halved and subtracted from 1 to convert the dissimilarity into a similarity score between 0 and 1.

The right-hand side of the equation is then used to weight the similarity score to reflect our confidence. As the proportional similarity score is calculated on a limited subset of flows from each cluster (flows in shared bags), the average number of data points used from each cluster weights the score. This guarantees that two clusters can only be similar when shared bags largely describe them both. In turn, when the shared bags explain only a small portion of one or both clusters, the similarity is weighted lower to reflect low confidence. This weighting is used by the similarity function to prioritise reliability.

6.3.3 Unsupervised Flow Cluster Aggregation

Our approach to flow cluster aggregation is presented in Algorithm 6.1. The algorithm is inspired by classic hierarchical agglomerative cluster aggregation strategies [54], but is more aggressive. Specifically, classic agglomerative clustering algorithms only merge the single most similar pair of clusters per iteration. Instead, our algorithm allows multiple aggregations per iteration. The intuition behind this stems from our proportion-based similarity metric described in the previous section. As cluster aggregation naturally changes bag proportions, two initially similar clusters can become relatively dissimilar over time. To prevent missing similar pairs, we thus allow all pairs above a threshold to be aggregated in each loop. To prevent greediness, the algorithm initialises with the strictest similarity threshold of 1. After each iteration, the threshold steps towards a user-defined minimum and iteratively works until convergence. An added benefit of this is the tendency for the algorithm to converge in fewer iterations than classic aggregation techniques.

```
Input:  set of clusters K trained on flows X = {x_i | i = 1, 2, ... n};
        similarity threshold s_t; bag reduction threshold p;
        aggregation rate step
Output: set of clusters S where |S| ≤ |K|

1: CCC(K, X, p)
2: S = K, t = 1
3: Repeat until convergence:
4:      For i = 1 ← |S|:
5:              For j = i + 1 ← |S|:
6:                      merge i, j in S if Similarity(i, j) >= t:
7:      t = t - step if t > s_t else s_t
8: CCC(K, X, 1)
```

Algorithm 6.1. Algorithm for Reliable Unsupervised Flow Cluster Aggregation

CCC is used both before and after the cluster aggregation. Prior to aggregation, the bag reduction is performed with a user-defined threshold parameter p, as described in Equation 15. This parameter dictates how aggressively bags are merged. This reduction increases overlap in bags between clusters, which intuitively allows improved aggregation. Reducing the bags post-aggregation is conducted with a threshold = 1. As the bags have already been reduced once and the clusters aggregated, we propose using the highest threshold for maximum reliability. The final set of bags can then be paired with the aggregated cluster set for analysis.

One final point on the algorithm is the use of logical equivalence. Due to the allowance of many aggregated pairs per iteration, situations frequently arise where some cluster A is similar to cluster B and cluster B is similar to cluster C, but A is *not* similar to C. We again use the core equivalence rule in Equation 16 to resolve such situations, and hence merge all three clusters.

6.4 Experimental Evaluation

In this section, we present the results of our approach. We evaluate primarily against the pre-aggregated cluster models to focus on both the relative reliability of reduced cluster sets as well as the value gained from the process.

6.4.1 Experimental Setup

The clusters evaluated in this section are generated by k-Means on the *wide* dataset (Chapter 3.1). We use k = 500 clusters for relatively pure representations of the 7 traffic classes in the dataset. We evaluate two components of the system in terms of overall accuracy. First, we evaluate the reliability of using Cross-Cluster Correlation under different correlation thresholds. We then use held-out samples of test data to evaluate the classification effectiveness of aggregated clusters alongside the final set of bags against the original, un-aggregated model. To identify each cluster, we use full class labels to find the most common class.

6.4.2 Effectiveness of Cross-Cluster Correlation

In Table 6.1, the effects of varying the correlation threshold p in CCC are presented. When $p > 1$, this is equivalent to the initial bag-of-flows model. The initial model comprised 47,395 bags that described over 400,000 flows in the dataset. Each bag was 100% accurate at describing a single class.

Table 6.1: Number of bags and bag accuracy under different thresholds in Cross-Cluster Correlation.

Correlation Threshold p	Number of Bags	Bag Accuracy
$p > 1$ (No Bag Reduction)	47395	100%
$p = 1$	4674	99.86%
$p = 0.75$	4500	99.86%
$p = 0.5$	2612	99.66%
$p = 0.25$	141	25.39%

When $p = 1$, the number of bags was reduced by over 90% to 4674 bags. 99.86% of the flows in these bags correctly corresponded to a single application. The same accuracy was observed for $p = 0.75$, which had a further reduction of 174 bags. Much more significant reduction was found at $p = 0.5$. With 2612 bags, this setting resulted in just over half the bags from the higher settings. The cost of this was just 0.20% lower accuracy, caused by the aggregation of very small and impure bags. At $p = 0.25$, the error introduced is much more significant. At just 141 bags, the low threshold resulted in just over 25% bag accuracy. Under this setting, bags rarely need to co-occur to be deemed equivalent. Due to occasional cluster impurities and the logical equivalence rule used during bag merging, such a low threshold greedily aggregates.

6.4.3 Effectiveness of Cluster Aggregation

We vary the similarity threshold s_t from 0.9 to 0.5 and evaluate the impact it has on cluster aggregation. $s_t = 0.5$ is selected as a logical minimum similarity threshold to ensure a majority of flows in the candidate pair of clusters is considered before merging. Bags are reduced prior to clustering with CCC threshold $p = 0.5$. We use a moderate *step* = 0.10 and reduce with CCC again after aggregation using a threshold of $p = 1$ to obtain the final number of bags. Table 6.2 presents the number of clusters and classification accuracy, as well as the final number of bags and bag accuracy under different threshold settings.

Table 6.2: Effectiveness of flow cluster aggregation under different similarity thresholds.

Aggregation	Clusters	Classification Accuracy	Number of Bags	Bag Accuracy
No Aggregation	500	93.98%	2612	99.66%
$s_t = 0.9$	411	93.98%	2555	99.65%
$s_t = 0.7$	282	93.98%	2305	99.62%
$s_t = 0.5$	126	93.00%	1562	99.36%

With a high similarity threshold of 0.9, the initial set of clusters was reduced from 500 to 411. The cluster aggregation was done with perfect aggregation accuracy, i.e. the most-common class in the aggregated clusters fully matches the majority classes in the original set. These 411 clusters reduced the number of bags by 57, and did so with a loss of just 0.01% bag accuracy. The mid-range similarity threshold of 0.7 reduced the initial number of clusters to just over half. Again, the aggregated clusters performed equivalently to the original cluster set in terms of accuracy. This setting also further reduced the total bag count by 250, with just 0.04% lower bag accuracy compared to the un-aggregated model. The lowest similarity threshold of 0.5 reduced the cluster set to just 126 clusters, and did so at a minor loss of 0.98% classification accuracy. The low similarity threshold also significantly reduced the number of bags to a total of 1562. These bags were again extremely accurate at 99.36%, a total loss of just 0.30%. Of the 126 clusters in this setting, 86 are observed to belong to the HTTP class. The other

six traffic classes make up the remaining clusters, with four for FTP, three for BitTorrent, six for HTTPS, seven for SMTP, and two for SSH. The final 18 clusters belong to the DNS class.

6.5 Analysis and Discussion

The significantly reduced number of clusters and aggregated bags are the two key factors that demonstrate both the potential and value our system. Under a moderate similarity threshold, the number of clusters was approximately halved with no loss in accuracy. Even under the lower similarity thresholds, clusters were found to aggregate reliably for the traffic classes in this dataset. We attribute the reliability to both CCC and our similarity function.

Bag merging with CCC is demonstrated to not only be very reliable, but also valuable for the aggregation process. With our reliability-oriented similarity function, high similarity scores are difficult to obtain when the number of bags is large. Accurately reducing this number allowed notably more overlapping bags between clusters. Furthermore, similar proportions were more common. Even the strictest (and hence most reliable) CCC threshold setting considerably reduced the number of bags to fewer than 10% of the original bag-of-flows model. Moderate and higher thresholds naturally reduced the number of bags further, but also did so with very good reliability. Due to expected purity in traffic flow clusters, *any* majority overlap between two bags was seen to be a good method of determining equivalence. CCC did perform notably poorly with a low equivalence threshold, but this result is unsurprising when considering the greedy nature of logical equivalence. While average cluster purity was typically high, we identified cases where the most common class accounted for as low as 35% of flows in a cluster. In such cases, low thresholds allow these infrequent co-occurrences in impure clusters to claim equivalence. These errors are then propagated due to logical equivalence rules. However, we reason that moderate to high thresholds are safe choices for reliable bag aggregation in any system with reasonably pure clusters.

The success of our similarity function is attributed to careful consideration of impure clusters. We found numerous situations where cluster impurity demonstrated the value of a proportion-based similarity metric. For example, we frequently observed FTP clusters containing a small but notable portion of SMTP flows. We then found SMTP clusters with similar portions of FTP flows. These clusters could share identical bags, and the clusters could be fully described by the same set of bags. The proportional differences in these bags were therefore critical for distinguishing the classes. Furthermore, the confidence-based similarity weighting causes the function to be very particular about assigning high scores. The consequence is that two clusters that truly belong to the same class may have very poor similarity scores when few of their bags are shared. This explains why the final cluster

sets were still reasonable large, even under low similarity thresholds. However, the critical benefit the similarity function is that it prioritises reliability, and is not expected to often give high scores to different classes. The strictness and reliability of this function affords the use of our more aggressive cluster aggregation algorithm, as the strictness and aggressiveness balance one another.

As was mentioned earlier, the strict similarity function comes at the cost of the overall size of cluster reduction. At approximately a quarter of the starting value, our best aggregation converged at 126 clusters and was a clear improvement over the initial setting. However, this is still a relatively large number compared to the seven traffic classes. We found that all classes aggregated reasonably well except for the HTTP class. Making up 86 of these 126 clusters, the majority of these HTTP clusters did not aggregate whatsoever. We attribute the cause of this to a combination of the strict similarity function and generally less overlap in bags between HTTP clusters. The lower correlation of the HTTP bags between clusters indicates that this class requires a lower threshold to aggregate well. However, the cost of lower thresholds would be instability for the remaining classes. If cluster purity can be reliably improved, lower thresholds can then be used to further cluster reduction. The value of targeting these less-correlated classes is left to future work. We also leave the automatic selection of good thresholds to future work.

Despite the relatively large number of resultant clusters, our approach still offers important value towards improving the interpretation of unsupervised flow models. Previous systems identify clusters through manual analysis, usually by labelling only cluster centroids or by sampling and identifying a sample of flows per cluster. Considering that we observed clusters with as low as 35% purity, these approximation methods are imprecise. Our system can improve interpretability in a few key ways. First, with significantly fewer clusters, we naturally require fewer labels to identify the system. To counter impreciseness, we can exploit our large flow bags. Labelling a single flow from each cluster's largest bag requires at most one label per cluster, yet our approach can label confidently due both bag equivalence and the size of our bags. As bags are present in multiple clusters, careful selection of bags can identify the system in even fewer labels. Second, CCC and bag aggregation can also *fully* label the system with just one label per bag. Our smallest final number of bags was 1562, and these bags were 99.36% correct. Labelling a single flow per bag would thus label our entire system (and dataset) using approximately 0.38% of its flows. It would do so with over 99% accuracy. Our aggregation approach can therefore be used as a pre-processing step for supervised learning approaches. Supervised approaches are the most effective for traffic classification [18]-[22], yet their requirement of fully labelled data is often considered unrealistic. As network flows generally constitute big data [10],

carefully selecting the flows to label with our approach can thus afford simpler and much more realistic uses of supervised algorithms. Future work can explore the effectiveness of supervised learning systems trained under this model.

Again, due to big data expectations, the time complexities for our cluster aggregation and CCC techniques are crucial. Assuming careful implementation of cluster bag memberships, cluster aggregation can be achieved using our algorithm with a time complexity of $O(Tk^2)$, where k is the number of clusters and T is the number of iterations until convergence. We observed $T \approx ceil\left(\frac{1 - aggregation\ threshold}{step\ size} + 1\right)$ due to the aggressive aggregation. Using an initial $k \approx 500$, the algorithm can therefore be seen as sufficiently efficient in the traffic flow domain. Given that there are $\frac{b\,(b-1)}{2}$ unique pairs of bags, bag merging through CCC is a $O(b^2)$ operation. In an n flow dataset, $b \leq n$. As n is typically very large, bag reduction is unrealistic in a worst-case bag-of-flows model (that is, $b \approx n$). However, in practice we find $b << n$, making the algorithm reasonable for large-scale datasets.

6.6 Conclusions

This chapter introduced an approach to address the main drawback of existing unsupervised flow classification approaches using exclusively flow correlation information. Through a fully original system, both the potential and value of reliable cluster and flow aggregation in network traffic classification are exhibited. Our approach combines a proposed flow correlation property and a cluster similarity function to significantly reduce the number of variables in a system. From our results, we reasoned how aggregated clusters and flow groups can be exploited to simplify the interpretation of unsupervised traffic flow models, and to do so with increased confidence. Furthermore, we proposed how our approach can be used as a pre-processing step for realistic uses of supervised algorithms by labelling just 0.38% of our dataset. Future work could consider a wider range of traffic classes and a methodology for reliable parameter selection.

Chapter 7
Conclusions and Future Directions

Network traffic classification is a fundamental task at the core of many network security, management, and surveillance functions. Machine learning techniques can succeed even when traditional approaches would fail, and have thus demonstrated strong potential for the traffic classification problem. In Chapter 4, we analysed multiple traffic flow formats and concluded that certain formats are more appropriate for use in statistical traffic classification models. In Chapter 5, we proposed a number of innovations to improve existing semi-supervised flow classifiers. Our approach uses minimal pre-labelled training data and leverages both cluster impurity and unlabelled flows to reliably and effectively classify both known and unknown classes of traffic. Our approach was shown to outperform comparable and leading semi-supervised approaches by upwards of 16%. Furthermore, our approach was shown to be the most stable, and classifies significantly faster than the comparable approaches. Finally, in Chapter 6 we proposed an original framework for simplifying the interpretation of unsupervised traffic flow models. Since these models typically have many more clusters than traffic classes, our system combines cluster aggregation and flow aggregation to significantly reduce the complexity of unsupervised systems. We showed how our system can be more confidently identified with as low as one quarter of the effort required by classic unsupervised approaches, and how our system can be fully labelled by identifying a minimal fraction of the dataset.

We propose two key areas of focus for future directions. First is the accommodation of changing traffic patterns and emerging applications. As networks change over time, it is reasonable to expect that the statistical properties of existing classes change also. Furthermore, it is also expected that new applications emerge, while existing classes may grow in significance or be phased out. While the approaches we considered throughout this thesis effectively use statistical properties to distinguish traffic classes, they require intermittent re-training to maintain relevance and effectiveness. In future works, accommodating change through incremental learning strategies is key to implementing real-world systems. These strategies can be extended to model changes in application volumes over time, and therefore efficiently predict growth and decline of classes for efficient management and decision-making. The second future direction we propose is to expand the system presented in Chapter 6. The proposed system demonstrates strong potential and value for cluster aggregation in unsupervised flow models. As a natural counter to the big data challenges presented by network traffic, furthering the reality of unsupervised approaches through increased model simplicity has extremely important implications for the future of practical statistical traffic classification.

References

[1] T. Karagiannis, B. Andre, and F. Michalis, "Transport layer identification of P2P traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 121-134.

[2] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semi-supervised network traffic classification," in *ACM SIGMETRICS Performance Evaluation Review*, 2007, vol. 35, p. 369.

[3] S. Zander, T. Nguyen, and A. Grenville, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, 2005, pp. 250-257.

[4] J. Ma, L. Kirill, C. Kreibich, S. Savage, and G. Voelker, "Unexpected Means of Protocol Inference," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 313-326.

[5] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding? [p2p traffic measurement]," in *Global Telecommunications Conference, 2004. GLOBECOM'04*, 2004, vol. 3, pp. 1532-1538.

[6] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194-1213, Oct. 2007.

[7] N. Williams, S. Zander, and A. Grenville, "Evaluating machine learning algorithms for automated network application identification," Center for Advanced Internet Architectures, Melbourne, VIC, Tech. Rep. B 60410, 2006.

[8] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. Vasilakos, "An Effective Network Traffic Classification Method with Unknown Flow Detection," in *IEEE Transactions on Network and Service Management*, 2013, pp. 133-147.

[9] S. Marchal, X. Jiang, R. State, and T. Engel, "A Big Data Architecture for Large Scale Security Monitoring," in *2014 IEEE International Congress on Big Data*, 2014, pp. 56-63.

[10] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," in *IEEE Communications Surveys & Tutorials*, 2014, vol. 16, no. 4, pp. 2037-2064.

[11] S. Zander, T. Nguyen, and G. Armitage, "Self-learning IP traffic classification based on statistical flow characteristics," in *International Workshop on Passive and Active Network Measurement*, 2016, pp. 324-328.

[12] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, 2005, pp. 250-257.

[13] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classification Using Clustering Algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, 2006, pp. 281-286.

[14] J. Erman, A. Maganti, and M. Arlitt, "Internet Traffic Identification using Machine Learning," in *Proceedings of the 49th IEEE GLOBECOM. San Francisco*, 2006, pp. 1-6.

[15] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. Yang, "Internet Traffic Classification Using Constrained Clustering," in *IEEE Transactions on Parallel and Distributed Systems*, 2014, pp. 2932-2943.

[16] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," in *Journal of Computer and System Sciences*, 2013, vol. 79, no. 5, pp. 573-585.

[17] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. Zhang, "A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks," in *ACM Transactions on Knowledge Discovery from Data*, 2012, pp. 1-34.

[18] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," in *ACM SIGCOMM Computer Communication Review*, 2006, pp. 5-16.

[19] T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic," in *IEEE/ACM Transactions on Networking*, 2012, vol. 20, no. 6, pp. 1880-1894.

[20] L. Bernaille and R. Teixeira, "Early Recognition of Encrypted Applications," in *International Conference on Passive and Active Network Measurement*, 2007, pp. 165-175.

[21] L. Bernaille and R. Teixeira, "Early Application Identification", in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, p. 6.

[22] A. Fahad, K. Alharthi, Z. Tari, A. Almalawi, and I. Khalil, "CluClas: Hybrid clustering-classification approach for accurate and efficient network classification," in *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 168-176.

[23] F. Qian, G. Hu and X. Yao, "Semi-supervised internet network traffic classification using a Gaussian mixture model," in *AEU - International Journal of Electronics and Communications*, 2008, vol. 62, no. 7, pp. 557-564.

[24] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," in *IEEE/ACM transactions on networking 23*, 2015, pp. 1257-1270.

[25] Cisco Systems, "Cisco IOS NetFlow," 2016. [Online]. Available: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html. [Accessed: 19- Jul- 2016].

[26] M. Soysal and E. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451-467, Jun. 2010.

[27] M. Tavallaee, W. Lu, and A. Ghorbani, "Online classification of network flows," in *Communication Networks and Services Research Conference, 2009*. 2009, pp. 78-85.

[28] C. Rotsos, J. Van Gael, A. Moore, and Z. Ghahramani, "Probabilistic Graphical Models for Semi-Supervised Traffic Classification," in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, 2010, pp. 752-757.

[29] M. Najafabadi, T. Khosgoftaar, C. Kemp, N. Seliya, and R. Zuech, "Machine learning for detecting brute force attacks at the network level," in *IEEE International Conference on Bioinformatics and Bioengineering (BIBE) 2014*, 2014, pp. 379-385.

[30] Mawi.wide.ad.jp, "MAWI Working Group Traffic Archive," 2016. [Online]. Available: http://mawi.wide.ad.jp/mawi/. [Accessed: 27- Nov- 2015].

[31] D. Arndt, "NetMate Meter," 2016. [Online]. Available: http://sourceforge.net/projects/netmate-meter/. [Accessed: 17- Mar- 2016].

[32] WAND Network Research Group, "WITS: Waikato Internet Traffic Storage," 2016. [Online]. Available: http://wand.net.nz/wits/. [Accessed: 11- Sep- 2016].

[33] S. Alcock, P. Lorier, and R. Nelson, "Libtrace: A Packet Capture and Analysis Library," in *ACM SIGCOMM Computer Communication Review*, 2012, vol. 42, no. 2, p. 42.

[34] Internet Engineering Task Force, "RFC 7011 - Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," 2013. [Online]. Available: https://tools.ietf.org/html/rfc7011. [Accessed: 19- Jul- 2016].

[35] AARNet, "Australian Academic and Research Network," 2016. [Online]. Available: https://www.aarnet.edu.au/. [Accessed: 27- Jul- 2016].

[36] SourceForge, "NFDUMP," 2016. [Online]. Available: http://nfdump.sourceforge.net/. [Accessed: 27- Jul- 2016].

[37] J. Steinberger, L. Schehlmann, S. Abt, and H. Baier, "Anomaly Detection and mitigation at Internet scale: A survey," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2013, pp. 49-60.

[38] C. Bucci, W. Song, C. Feng and S. Sharma, "Portable system for monitoring network flow attributes and associated methods", U.S. Patent 9 344 344, May 17, 2016.

[39] C. Gao, W. Ding, Y. Zhang, and J. Gong, "Estimating Average Round-Trip Time from Bidirectional Flow Records in NetFlow," in *Recent Progress in Data Engineering and Internet Technology*, 2012, pp. 415-423.

[40] L. Maaten and G. Hinton, "Visualizing data using t-SNE," in *Journal of Machine Learning Research*, 2008, pp. 2579-2605.

[41] L. Wang, U. Nguyen, J. Bezdek, C. Leckie, and K. Ramamohanarao, "iVAT and aVAT: enhanced visual analysis for cluster tendency assessment," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2010, pp. 16-27.

[42] T. Havens and J. Bezdek, "An Efficient Formulation of the Improved Visual Assessment of Cluster Tendency (iVAT) Algorithm," in *IEEE Transactions on Knowledge and Data Engineering*, 2012, pp. 813-822.

[43] J. Huband, J. Bezdek, and R. Hathaway, "bigVAT: Visual assessment of cluster tendency for large data sets," in *Pattern Recognition*, 2005, vol. 38, no. 11, pp. 1875-1886.

[44] D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. Bezdek, and T. Havens, "clusiVAT: A mixed visual/numerical clustering algorithm for big data," in *2013 IEEE International Conference on Big Data*, 2013, pp. 112-117.

[45] Scikit-Learn, "Ensemble methods," 2016. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html. [Accessed: 01- Mar- 2016].

[46] L. Breiman, "Random forests," in *Machine learning*, 2001, vol. 45, no. 1, pp. 5-32.

[47] C. Strobl, A. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," in *BMC bioinformatics*, 2007, p. 1.

[48] P. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Boston, MA: Pearson, 2005, pp. 158-164.

[49] B. Menze, B. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. Hamprecht, "A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data," in *BMC Bioinformatics*, 2009, vol. 10, p. 213.

[50] L. Breiman, "Out-of-bag estimation," Statistics Department, University of California Berkeley, Berkeley, CA, Tech. Rep. B 33 34, 1996.

[51] E. Boschi and B. Trammell, "Bidirectional Flow Measurement, IPFIX, and Security Analysis," in *FloCon 2006*, 2006, pp. 1-4.

[52] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network Traffic Classification Using Correlation Information," in *IEEE Transactions on Parallel and Distributed Systems*, 2013, pp. 104-117.

[53] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in *IEEE Communications Surveys & Tutorials*, 2008, vol. 10, no. 4, pp. 56-76.

[54] J. Ward, "Hierarchical Grouping to Optimize an Objective Function," in *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236-244, Apr. 1963.