2020-01-01

Evaluating Flow Features for Network Application Classification

Carlos Alcantara
University of Texas at El Paso

EVALUATING FLOW FEATURES FOR NETWORK

APPLICATION CLASSIFICATION

CARLOS ALCANTARA

Master's Program in Computer Engineering

APPROVED:

Michael P. McGarry, Ph.D., Chair

Patricia Nava, Ph.D.

Eric Smith, Ph.D.

Stephen Crites, Jr., Ph.D.
Dean of the Graduate School

EVALUATING FLOW FEATURES FOR NETWORK APPLICATION CLASSIFICATION

by

CARLOS ALCANTARA, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2020

Acknowledgements

I wish to acknowledge my thesis advisor Dr. Michael P. McGarry of the Electrical and Computer Engineering department of the University of Texas at El Paso. I am thankful for his mentorship and guidance not only during the completion of this work, but throughout my academic career. The time and effort dedicated to assist me has never gone unnoticed and is deeply appreciated. Thank you for always being available and willing to help. I am a better student, researcher and professional because of him.

I would also like to thank my colleague Christopher A. Mendoza for his help and insight on the work conducted in the lab. He is a great researcher and I am glad I had the opportunity to have worked with him.

I am deeply thankful for my parents and sister. None of my success would be possible if it wasn't for their love and support. Thank you for all that you constantly do to enable me to pursue my goals.

And finally, I am extremely grateful for my girlfriend. She has always encouraged and assured me that I would succeed. Words cannot express how appreciative I am of her unconditional support and encouragement through this and all my endeavors. I am truly fortunate to have such a wonderful woman by my side.

Abstract

Communication networks provide the foundational services on which our modern economy depends. These services include data storage and transfer, video and voice telephony, gaming, multimedia streaming, remote invocation, and the world wide web. Communication networks are large-scale distributed systems composed of heterogeneous equipment. As a result of scale and heterogeneity, communication networks are cumbersome to manage (e.g., configure, assess performance, detect faults) by human operators. With the emergence of easily accessible network data and machine learning algorithms, there is a great opportunity to move network management towards increasing automation. Network management automation will allow for a reduced likelihood of human error in network configuration, improved productivity from network managers as redundant tasks are automated, simplified scalability, and greater insight into network operation. Network application classification, the process of identifying the network application associated with trains of packets called flows, is a critical task in the automation of network management. This association of network applications with network traffic is critical for improving network management as it will allow setting application-specific policies to optimize network operation, enhancing security measures by blocking certain applications with improved firewall configurations, and developing a more reliable quality of service by prioritizing time-sensitive applications.

This work studies the classification performance of a basket of network flow features. We utilized three categories of flow features: inherent, derived, and engineered. In our first experimental analysis, we set out to uncover the inherent and derived feature's ability to classify network flows. We developed an expert system to generate application labels to serve as training data, which is used to train our models on two inherent and one derived feature. Flows are analyzed by implementing three supervised machine learning techniques for classification: k-nearest neighbors, decision trees and random forest. These experiments varied the number of applications and type of flows, all or only large, in a traffic data capture

from UKY's university network. For our subsequent experimental analysis, we engineered three flow features based on host behavior presented by the authors of BLINC and examined their influence on traffic classification performance when combined with the features from the previous experiments. A new UKY data set is captured using deep packet inspection to obtain training labels and the same three machine learning techniques are employed. In these subsequent experiments, we varied the set of features used for classification by always including the three inherent and derived features and one combination of adding the three engineered features. Our initial experiments reveal that the inherent and derived features can adequately classify a subset of applications while focus on large flows slightly reduces performance. Our subsequent experimental analysis concludes that the use of engineered features provides a statistically significant improvement on classification performance for decision tree and random forest, while KNN is most effective with only the original three inherent and derived features.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Communication networks are the pillar holding up our modern civilization. The creation of communication networks has transformed the world by allowing the development of a global society while redefining many aspects of our lives in the process. First, communication networks have reshaped our idea of community. Traditional communities bound by geographic location are no longer our only avenue for socialization. The capacity to interact with anyone across the world has developed virtual communities, where people with similar hobbies and interest can come together to find a sense of inclusion and belonging. Our professional environment has also been impacted by permitting communication and collaboration across the globe. This has allowed companies to operate through multiple branches located in every corner of the world while still focusing on a cohesive product. As for employees, it has given many the option to work from home. Finally, communication networks have enabled e-commerce, where we can buy and sell goods and services from anywhere in the world. Whether its something as simple as a toothbrush or as significant as a car or home, we are able to make these purchases completely online. Anything that we can ever want or need is just a click away.

A clear example of just how truly vital communication networks have become to our society can be seen in the face of this COVID-19 pandemic. Communication networks have had an immediate impact as they have enabled the health industry to share information in real time to facilitate tracing the spread of the virus. A domino effect has also been felt on our day-to-day interactions with communication networks. First, the use of social media and news networks to disseminate information regarding the preventive measures, such as the stay-at-home orders and social distancing guidelines. These measures have

pushed society to find new ways to keep in touch with family and friends. Video and audio communication applications have become very useful tools for us to do so. Despite the pandemic, communication networks provide the infrastructure to allow many to continue to work from home and continue their education remotely. Tools like Blackboard Collaborate Ultra and Zoom, that allow for virtual classes and work meetings to take place, are the new norm for at least the near future. Finally, we have never been so reliant on e-commerce. Whether purchasing essentials, like groceries and take-out, or recreational items, like games or books to keep us busy in these difficult times, it is clear that communication networks have never been so necessary as they are today. Communication networks have been so ingrained in our daily routines, even before this pandemic, that it is nearly impossible to imagine a world without these services.

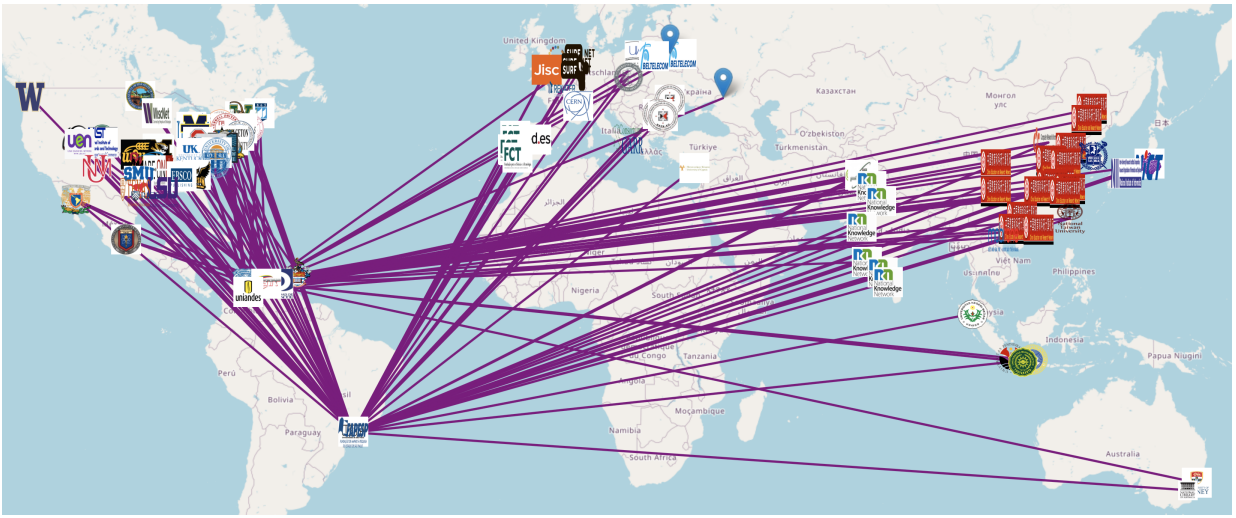


Figure 1.1: Map of autonomous systems communicating over the Internet [1].

A communication network is defined as an interconnection of computing devices with the purpose of sharing information. The largest and most obvious example of a communication network is the Internet, where network-enabled devices such as computers, smart phones and tablets are able to transmit images, audio, video and files over the network. The Internet is comprised of a series of interconnected networks. The networks that constitute the Internet are called autonomous systems (AS). Each AS in the Internet is assigned a

unique number for identification. For example, the University of Texas at El Paso (UTEP) is AS# 16461. Figure 1.1 is a visualization of the Internet, where each icon on the map represents an AS and the connecting purple lines correspond to the information transferred amongst these AS. Although this image represents a relatively small subset of Internet traffic, it provides a clear perspective of the global scale of communication networks and how information is shared across the world. Now, let us consider a topology of an AS such as the sample network of an educational institution as illustrated in Figure 1.2, where the variety of networking devices that are constantly interacting is exemplified. Core, layer 3, and layer 2 switches and routers connecting end users to file, web, and mail servers are just a few of the devices coordinating within an AS to provide communication services. This makes evident that a communications network can be a very complex system making network management a complicated task. The fact remains that none of the wonderful services mentioned earlier would be possible as they are today without communication networks. It is therefore in our best interest to make every effort so that these communication networks, that are so important to us, are operating as efficiently as possible. In order to do that, we must understand how to control, or manage, network operation.

Network management is best described using the International Telecommunications Union (ITU) M.3400 FCAPS model which delineates the responsibilities of network management as:

- **F**ault detection and correction
- **C**onfiguration and operation
- **A**ccounting and billing
- **P**erformance assessment and optimization
- **S**ecurity assurance and protection

For example, a network manager will account for the usage of its network in order to fine-tune the network configuration to provide optimal operation. Similarly, a new security

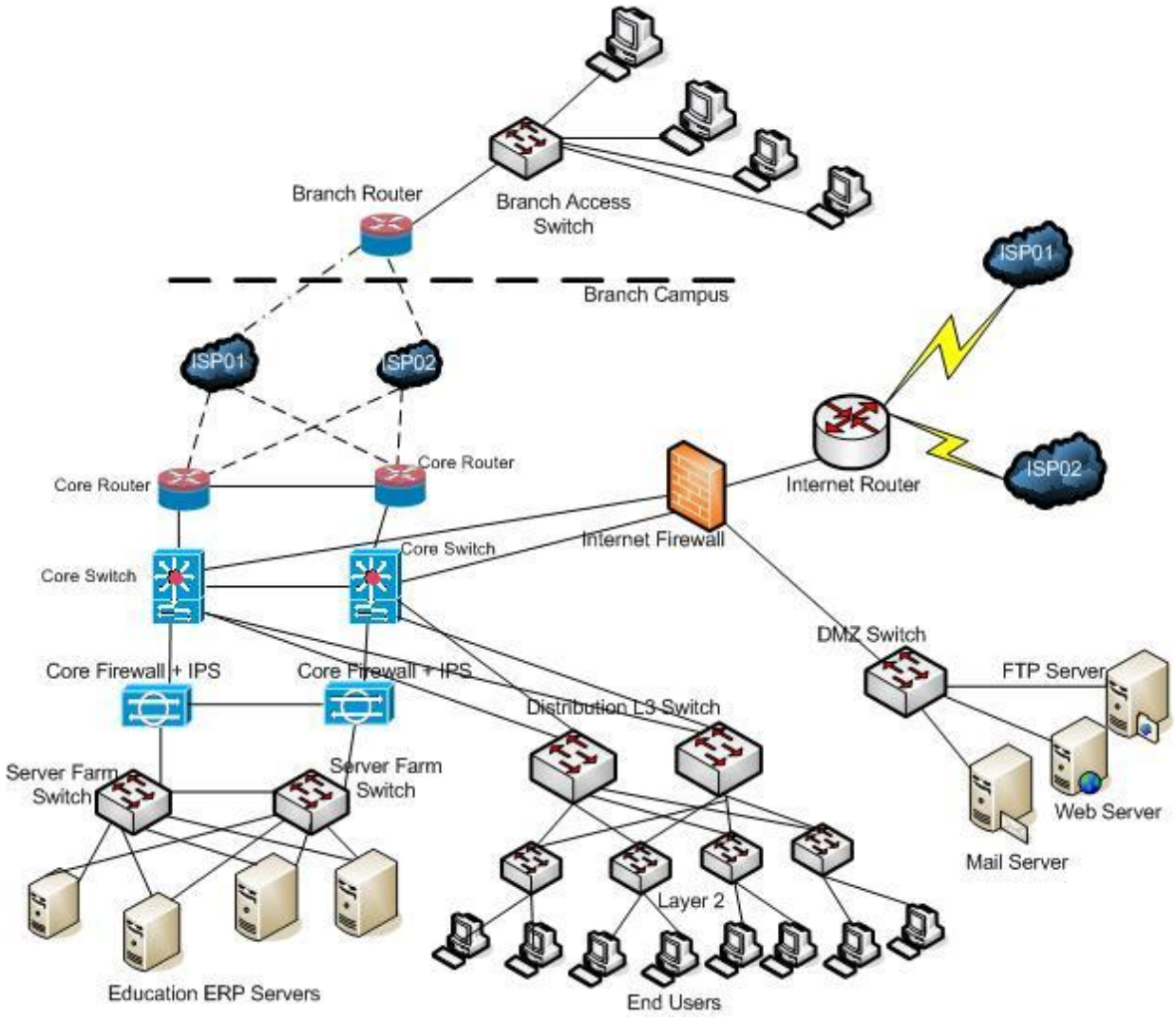


Figure 1.2: Network topology of a generic university AS [2].

policy is implemented by the network manager who then performs an assessment of network performance once this new policy is set. These responsibilities that define network management come with many challenges. There are four main challenges associated with network management. The first challenge is the scale of communication networks. This is not only in the sense that the Internet contains over 96,000 AS [3], but as Figure 1.3 illustrates, the size of a single AS can become overwhelming. The next challenge in communication networks is the heterogeneity of devices that constitute a network. Not only are there many types of devices that serve distinct functions within a network, but there are also different manufacturers with various models of these devices. Although these manufacturers design their equipment such that they are able to interoperate, most have proprietary software which indicates that the configuration of these devices is not uniform. This means that the network manager, when needing to update the Dell and D-Link switches along with the Linksys and Cisco routers to be able to access the new HP server, will need to write multiple versions of the same configuration file to update all these network devices. Now imagine having to check each of those distinct networking devices to determine which need the latest software update or a new security patch because of a new found vulnerability. Network management can quickly become a cumbersome task. The third challenge is the induction of human error. Whenever a human is interacting with any computing system there is a possibility of generating an error. Anyone who has ever written any computer code can attest to the devastating effects that a simple typo, such as a missing semicolon or closing bracket, can have on the functionality of a system. And finally, the last challenge is the frustration of a fault in network operation. The frustration can manifest in the end user detecting a misconfiguration and becoming upset with the network manager because the network that is so essential to them is not working as they expect. This can create further frustration for network managers as they are unable to detect and correct faults before the end users are affected. All of these challenges need to be handled by the network managers, who for the most part currently need to do all of this via a command line interface or SNMP. As is evident, there is a lot of work being asked of network managers.

In an effort to make network management a less burdensome task, there is a growing trend towards network management automation.

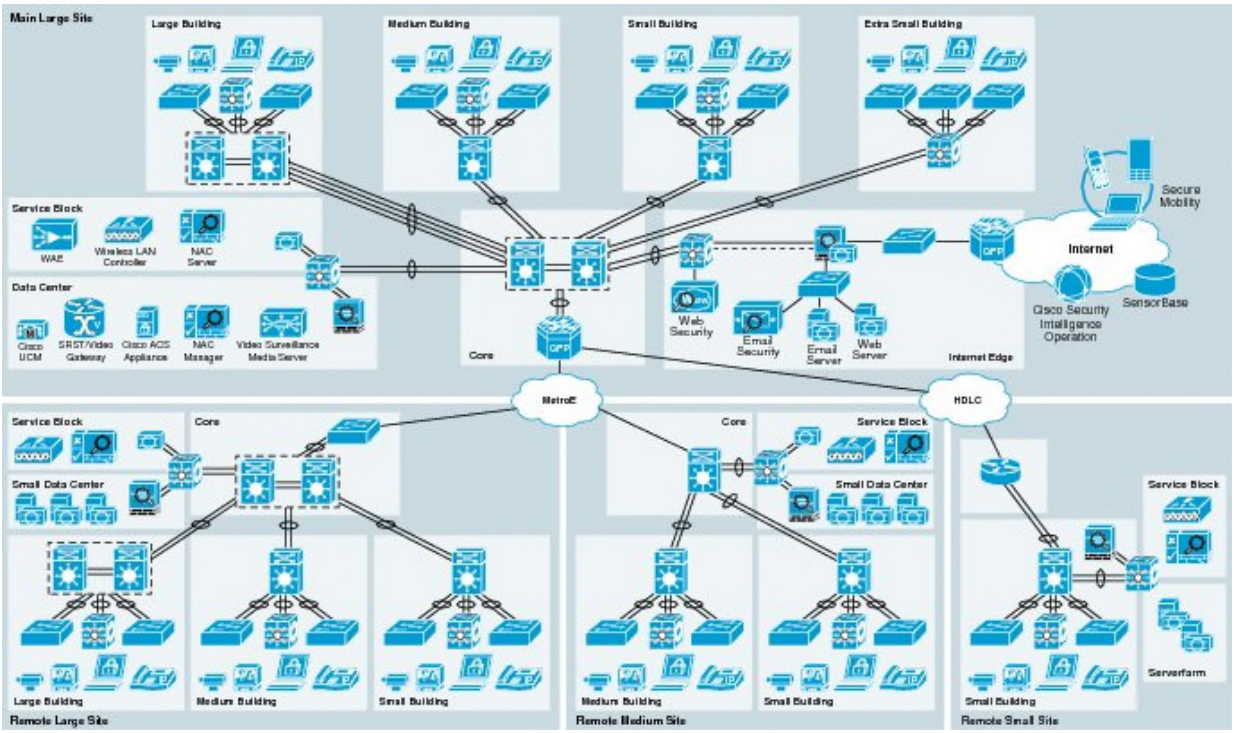


Figure 1.3: A network topology can become complex which makes them difficult to manage [4].

1.1 Automating Network Management

Network management automation is the process in which software is utilized to configure, provision, manage and test communication networks with minimal human assistance. The objective is not that of eliminating network managers from network management, but rather to limit the redundant and time-consuming tasks currently assigned to network managers by enabling these tasks to be performed automatically.

The introduction of software defined networking has opened the door for network management automation. Software defined networking (SDN) allows the separation of the data

and control planes in networking devices. While each networking device will control its data plane, SDN allows multiple network devices to be dynamically configured with software by a central controller. This means that to manage the network, changes are only made to the network controller which forwards these changes to all the networking devices within the network. Figure 1.4 provides a visual representation of SDN in comparison with traditional network management. This gives network managers simplified access to their network devices from a single location as opposed to connecting to each individual network device which becomes difficult to handle with the issues of scale and heterogeneity. Having a network that can autonomously detect and correct connection problems, self-optimize, or recognize and block security concerns such as cyberattacks would prove invaluable, and SDN gets us one step closer to achieving that goal. However, in order to make network management automation a reality, a deeper understanding of network utilization is necessary.

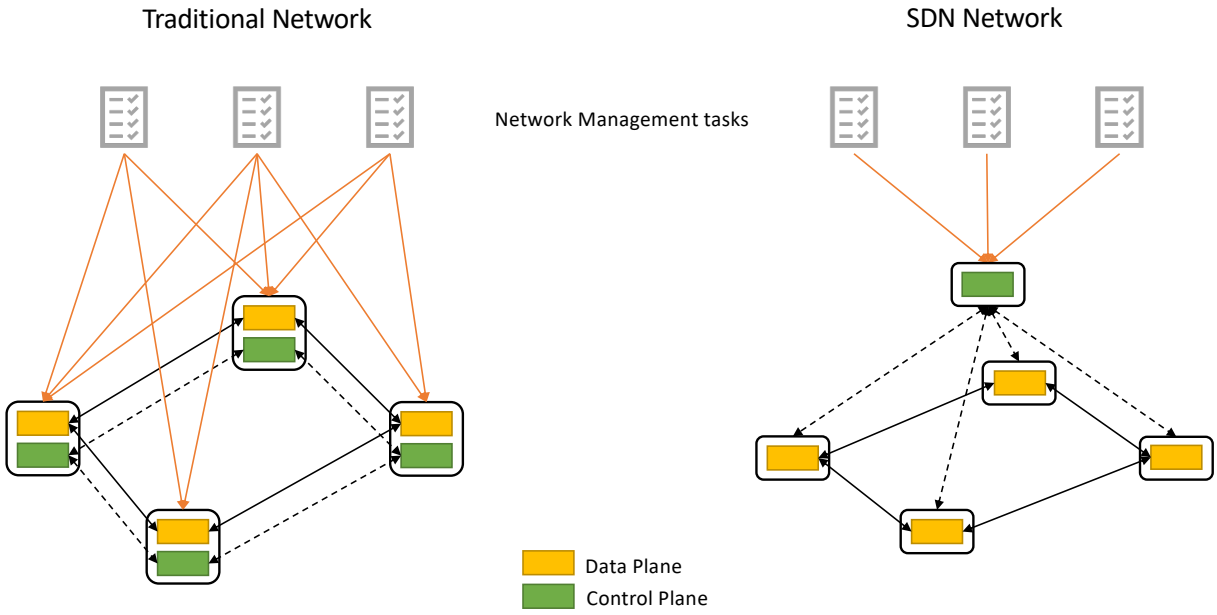


Figure 1.4: Software Defined Networking compared with traditional networking.

Let us envision a scenario where we have fine-grained knowledge of network behavior. This will allow application-specific configuration, where an application can be limited

to a percentage of network resources. For example, having an entire university network allocating all of its resources to video streaming is undesirable. By identifying traffic according to its generating application, a network manager can configure policies to limit this application to consume a third of the network resources. Understanding the performance of individual applications will also improve quality of service. By measuring latency of sensitive applications, such as live streaming, an Internet service provider can adjust the network as necessary to guarantee the service that their customers expect. Finally, recognizing how applications are communicating within a network over a period of time will allow for the generation of behavior patterns. These behavior patterns, represented by network flows, can then be used to detect faults by analyzing new network traffic with contextual anomaly detection techniques. The application that produced the network flows can serve as part of the context to detecting faults and misconfigurations before they affect the end users. The broad impact that network traffic classification by application will make on the understanding of network operation is the key to unlocking network management automation.

1.2 Network Application Classification

Network traffic consists of a series of data packets generated by a variety of applications and utilities propagating across a communications network. The process of applying a label to observed network traffic according to the program or process responsible for its creation is referred to as network application classification. The label applied is determined based on the characteristics of the network traffic (e.g. size, duration). This can be done at at the packet level, where each packet's generating application is identified, or at the flow level, where packets passing an observation point are aggregated based on similar characteristics, or features. Flows are created for the purpose of extracting a conversation communicating over the network as opposed to analyzing individual packets. Once flows are created, they are subsequently labeled according to the communicating application. The labels assigned

to the observed traffic can be coarse-grained such as peer-to-peer or bulk transfer, or fine-grained where the exact application which generated the message (e.g. BitTorrent, FTP) is identified.

There are three approaches to network application classification:

- Port number conventions - The transport layer protocol and port numbers of each packet are identified and the corresponding application registered to that combination is assigned as the generating application.

- Packet payload - The payload, which is the section of the packet where the actual message transmitted is stored, is compared to a set of patterns or signatures in order to identify the communicating application.

- Flow features - The characteristics generated during packet aggregation as well as information about the flow itself are used to infer the application generating the traffic. It is common for additional flow features to be created by combining information obtained during flow creation or by combining the information in the flows with outside data to develop supplementary features.

Although all three approaches have their advantages and disadvantages which are described in detail in Section 2.3, the port number convention's susceptibility to port abuse and inability to classify data on dynamic ports do not make it a feasible approach moving forward. Similarly, the packet payload's privacy concern along with the inability to classify encrypted data makes this approach unfit for the future as more applications are opting to encrypt their traffic. In contrast, the flow feature's wide deployment, respect of data privacy and ability to handle encrypted traffic make it the most adequate approach. The ability of the flow feature approach to classify network traffic is based on the capacity of the features generated during and after flow creation to uncover the communicating applications. Therefore, it is crucial that there is a meticulous evaluation of which features should be created and considered for network application classification when using the flow feature approach.

1.3 Thesis Outline

The inability for network managers to have a clear picture of how their network is being utilized makes network management automation an extremely difficult task. The reality is that network traffic classification is not a new problem, so it begs the question, why now? Well, there are several reasons for optimism. Recent improvements in computing systems allow for large volumes of data to be stored and analyzed quickly. This has kindled many efforts in making tools to analyze this data, such as machine learning and neural network open-source libraries, which have been made easily accessible to all. This has empowered researchers across all disciplines to leverage these tools in their efforts to solve problems specific to their domain.

Network traffic classification is a non-trivial problem with the potential to change the way in which networks are currently managed for the betterment of its users. This is a monumental task that is very actively researched by many across the world. Like many other complex problems, it is best to apply the "divide and conquer" approach where the focus is on a section of the problem and these solutions are built on top of one another as these sub-problems are solved. With this in mind, the focus of this work is in evaluating the performance of network flow features in network traffic classification. Specifically, the aim is to understand how different combinations of flow features are able to impact the ability for machine learning techniques to correctly predict the applications generating network traffic.

The rest of this work is organized as follows: Chapter 2 presents the definition of network flows, typical algorithms used for classification, approaches for network application classification and relevant work conducted on this topic. Chapter 3 describes the plan for the efforts conducted to evaluate the combination of flow features in classification performance, with results reported in Chapter 4. Finally, Chapter 5 provides concluding remarks as well as future work in this exciting and challenging problem.

Chapter 2

Background

Network traffic classification is a complex problem with many moving parts that come together in an effort to overcome this challenging hurdle in order to improve network management operation. Therefore, it is of great value to understand the individual parts involved in this problem separately before attempting to find an appropriate solution. The process of network traffic data capture and aggregation into flows is covered in section 2.1, typical algorithms used for data classification are presented in Section 2.2, the different types of data used in network traffic classification are described in Section 2.3 and finally current efforts are explored in Section 2.4.

2.1 Network Flows

2.1.1 Definition

According to [5] a flow is defined as a "set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties." These common properties are typically contained in packet headers. For example, source and destination IP addresses, source and destination port numbers, and information about the packet itself (e.g. transport layer protocol). The purpose of aggregating network packets into flows is to convert what appears to be an unordered set of packets communicating over a network into a set of meaningful information about these interactions without having to store all of the packets themselves. These interactions can provide an increased level of understanding of the network's behavior,

giving network managers, both human and automated, a better sense of how the network is being utilized.

This is analogous to what you may see at your local post office. Instead of network packets with IP addresses, the post office receives letters and packages with sender and delivery addresses. In this sense, letters and packages are grouped at the post office according to their destination. This enables the post office manager to allocate resources more efficiently. It would not make sense for a few letters to be sent across town using an airplane in the same way that it would not make sense for a university to allow most of its network to be used for playing video games online.

Because of the initial lack of a standard, most commercial networking hardware vendors created their proprietary flow export protocol, such as Cisco's NetFlow and Juniper's J-Flow. As a result of the growing inconsistency on the definition of a network flow, the Internet Engineering Task Force (IETF) created the Internet Protocol Flow Information eXport (IPFIX) protocol. This is the standardized protocol for flow export and is increasingly supported by most commercial vendors.

2.1.2 Creation

Flow creation can be considered in three distinct steps: packet observation, flow metering and export, and data collection.

Packet Observation

In this step packet data is read and routed for aggregation into a network flow. This data is captured from the communication network line by the Network Interface Card (NIC). This can be typically done in one of two ways: by connecting a capture device directly in-line, or by mirroring traffic from the line using forwarding devices onto the capture device. Once the packet is captured it receives a timestamp. Packet timestamps can be created with hardware, available on special NICs to increase accuracy, as well as software, which is how

most commodity cards perform timestamps.

Although this occurs for all captured packets, there are optional configurations to limit the amount of data and/or packets observed. Packet truncation at a predetermined packet length (snapshot length) can be selected to reduce the amount of data stored per packet as this can become resource intensive in large networks. Additionally, packet sampling and filtering, where a packet subset is selected according to sampling (e.g. 1 of every 50 packets) and/or filtering (e.g. packets with a certain size) configurations, can also be applied during the packet observation step [6].

Flow Metering and Export

Flow metering is the step in which observed network data packets are analyzed and aggregated into network flow records. This aggregation process is based on the Information Elements that define the data available in the flow records created. Information Elements (IEs) are the data fields that can be extracted from network traffic packets according to the IPFIX standard. Examples of IEs are source/destination IP address, source/destination port number, packet count, and byte count. The Internet Assigned Numbers Authority (IANA) is the entity tasked with maintaining a complete list of the standard IEs.

A flow table where all the information required for flow metering is referred to as the flow cache. In this cache, the IE data extracted from each network traffic packet is aggregated to an existing flow record or converted into a new flow record. These flow record entries in the cache will remain until it is deemed that the flow has completed. There are several specifications to determine if a flow has completed [6].

- Active Timeout - If the flow record is active for a prolonged period of time. This timeout allows to report activity of long-lived flows and is configurable by the network manager.

- Idle Timeout - No data for this particular flow record is received within the selected timeout. This is also configurable by the network manager.

- Natural Expiration - TCP packet with a FIN or RST flag is observed, which by TCP definition signals the end of the communication and thus the end of the flow. Figure 2.1 displays a message with natural expiration.
- Emergency Expiration - A number of flow records, configured by the network manager, are forced to expire if the flow cache becomes full.

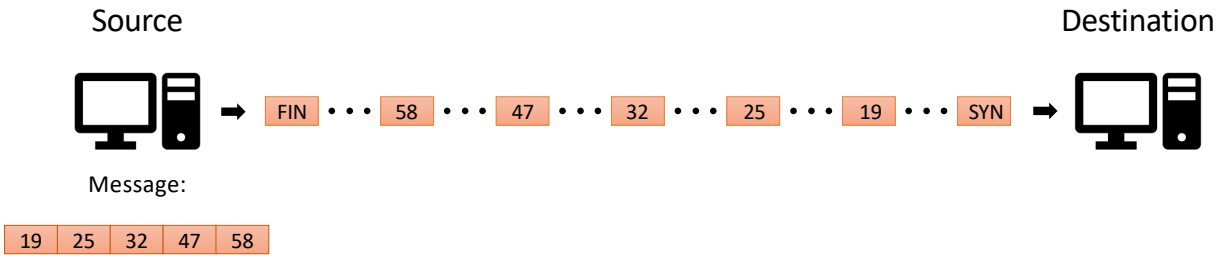


Figure 2.1: TCP message with SYN and FIN flags delineating message start and end.

Similarly to the packet capture stage, once a flow record completes it may go through a sampling/filtering stage. Once sampling and/or filtering is complete, an IPFIX message containing the completed flows is constructed. These messages are then sent, or exported, to the data collector.

The packet observation, flow metering and export steps are not always separated. In fact, these are often combined into a single device called a *flow exporter*. When the flow exporter is a dedicated device it is called a *flow probe*. Forwarding devices and firewalls are often already available in networks and many have flow export support, in which case no extra device is necessary and enabling the flow export function is simply a matter of configuration [6].

Data Collection

Network flow data collection is done within the flow collector. Here, the IPFIX messages are received and then written into storage devices. The flow collector can typically save the flow messages as flat files (e.g. binary or text files) as well as entries in a database (e.g.

MySQL). Flow collectors may conduct pre-processing tasks such as IP obfuscating, where the original IP addresses are altered or removed, as a security and privacy measure before saving the network flows [6].

Figure 2.2 provides a visual representation of the creation of network packets into flows. As traffic propagates through the network and into the destination host, packets are observed by the flow exporter. This exporter analyses the network packets and updates the flow cache table according to the observed traffic. Once the flow has been completely observed, such as the green packets in the figure, the flow cache entry is converted to an IPFIX packet which is sent to the flow collector. Finally, the IPFIX packet containing the flow information is stored, completing the creation of the flow representing the green traffic.

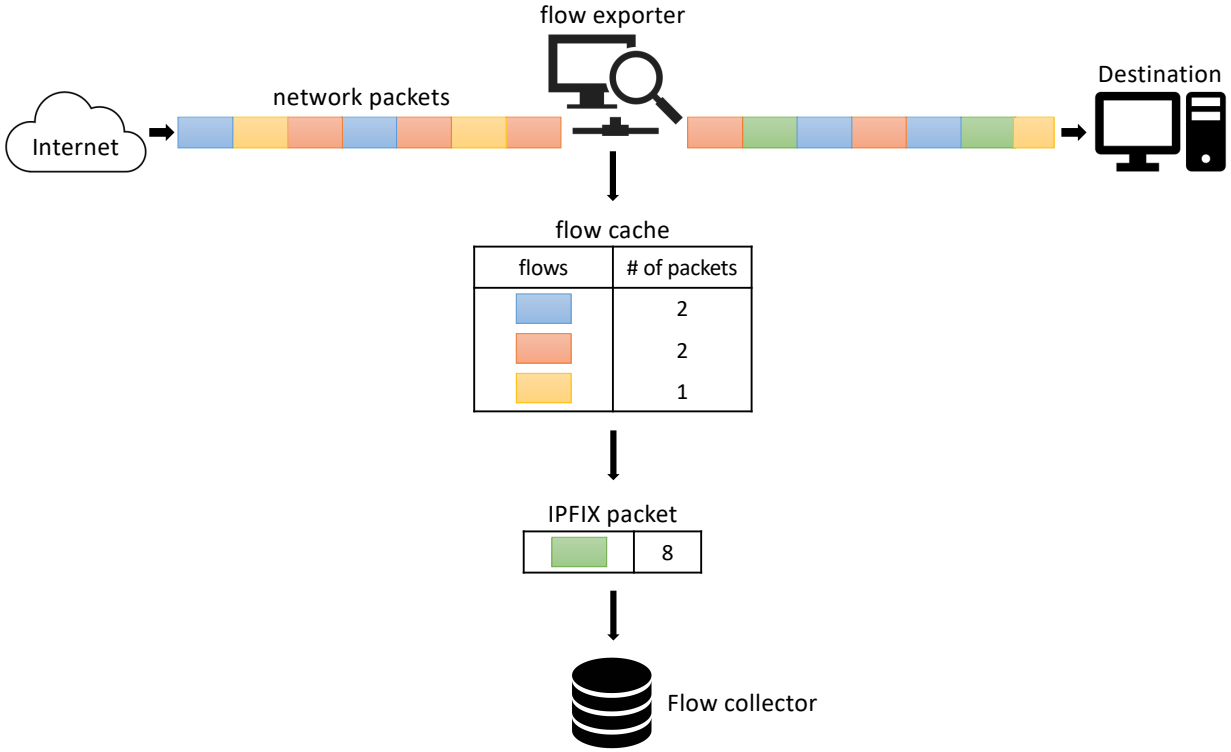


Figure 2.2: Network packet data is observed, exported and collected.

Once the data is collected, data analysis can take place. This can either be done manually or automated. Many companies provide products and services for automated

flow data analysis based on specific needs like security or performance monitoring.

2.1.3 Features

Although there is a predetermined number of IEs, or features, which a network flow object must include, it is by no means limited to only those specified by IEs. There are a myriad of features that can be included in a flow object definition but they will all fall into one of the three feature descriptions:

Inherent Features

The inherent feature are those that are obtained from the raw network traffic packet capture where no additional processing is needed to obtain these features. These are the features generated by the IEs in the flow metering process. This includes the standard five-tuple of features which are the minimum requirement for a network flow object construction:

1. Source IP address
2. Source port number
3. Destination IP address
4. Destination port number
5. Transport layer protocol

Additionally, any feature which can be read directly from the network packet such as:

- Start timestamp
- End timestamp
- Byte count
- Packet count

- Type of Service (ToS)
- TCP flags

Derived Features

Derived features are those that can be generated from the combination or analysis of the inherent features. For example, the duration of a flow can be determined from subtracting the start timestamp from the end timestamp, or the average number of bytes per packet can be determined from the byte count over the packet count.

Engineered Features

Engineered features are those that combine data from across a collection of flow objects and/or using additional data not contained in the network traffic packets. Appending geographic information such as source/destination country based on the IP addresses of each flow object, or including whether the flow occurred during the week or weekend are trivial examples of engineered features.

Feature engineering allows for creativity and innovation as there is an increasing amount of data in general being collected in all areas. This allows for information pertinent to a specific problem that would not be universally pertinent to other problems to be applied. The importance is in generating additional information that will provide an facilitate analysis of the problem at hand, in this case the classification of network traffic. One interesting approach is presented by [7] where network behavior is analyzed. This is further explored in Section 2.4.1.

2.2 Machine Learning Algorithms

Machine Learning is the study of algorithms and statistical models where inference and patterns are used to perform a specific task without explicit instructions. These algorithms

can be divided into four main categories:

- Supervised - Mapping a set of inputs to a set of outputs. The presence of sample data with the desired outputs is required in order to generate, evaluate and optimize the model's performance. A common example is an email spam filter. Email data with values such as sender, email length and time of day sent as well as the resulting label (spam or not spam) are presented for the model to "train" or find the patterns. Once the model has analyzed this data it can be used to predict on previously unseen email data.

- Unsupervised - Finding patterns and associations in data that has no specified output. In this approach the value lies in the algorithm revealing structure within the data. This method includes data clustering, anomaly detection and dimensionality reduction. An example of unsupervised learning is grouping customers by segments according to their spending habits which can be used to apply adequate marketing strategies to each group independently.

- Semi-supervised - A combination of using a small amount of labeled data with a large amount of unlabeled data to obtain the benefit of both supervised and unsupervised algorithms. This model is applicable where labeled data is scarce. This model could be used by labeling a small set of breast cancer scans and extrapolating those labels out to a much larger data set using an unsupervised method.

- Reinforcement - Reward based technique to encourage positive results and discourage adverse results. This model is common in games and decision based problems where there are defined states, such as the position of a game piece on a board, and the outcome can be quantified in win or lose, points total or other metrics where the model controls a system [8].

Because network traffic classification is by definition a supervised machine learning problem, our efforts are focused on this category.

2.2.1 Supervised Learning

Supervised learning consists of finding the relationship between a set of inputs (also called predictors, independent variables or features) and their corresponding output (also called responses, dependent variables or targets). The data for both the inputs and outputs can be quantitative (continuous) or qualitative (categorical). There are two distinct problem types that can be solved by supervised learning, regression and classification [9].

Regression

Regression problems are those where data is used to predict continuous outputs. Regression aims to find a model or function which represents the data with ratio or interval values.

In mathematical terms, regression finds the function that best predicts the true output y based on the estimation $\hat{y} = f(x)$ where $f(x)$ is the function based on the input data vector $x = [x_1, x_2, ..., x_n]$. Because there is a necessity to compare regression function models to find which best predicts the data, there needs to be a measure that evaluates how well the function predicts the output in comparison with the true output. This is done with a loss function which calculates the error in predictions.

Linear regression can provide a simple implementation of regression. Suppose height h is predicted as a function of weight w. Given a series of weight and height data pairs, linear regression will find a line that best fits the data by

$$\hat{h} = b_0 + b_1 w. \tag{2.1}$$

Because height and weight are not perfectly correlated, if they were there would be no need to calculate a prediction, there is always an error e between the optimal model's predicted height $\hat{h}$ and the actual height h given by the data. Note that because this error is a distance, the error must be the square root of the squared error

$$e = \sqrt{(h - \hat{h})^2} \tag{2.2}$$

to account for possible negative distances. b_0 and b_1 are the coefficients selected by the regression model based on the given data such that this error is minimized.

$$\min \sum^{\text{for all } h} e = \min \sum^{\text{for all } h} \sqrt{(h - (b_0 + b_1 w))^2}. \tag{2.3}$$

Once this model has been fitted with the appropriate b_0 and b_1, the model can be used to make height predictions based on any given value for weight. This can be expanded into a multidimensional linear model by changing equation 2.1 to a polynomial equation.

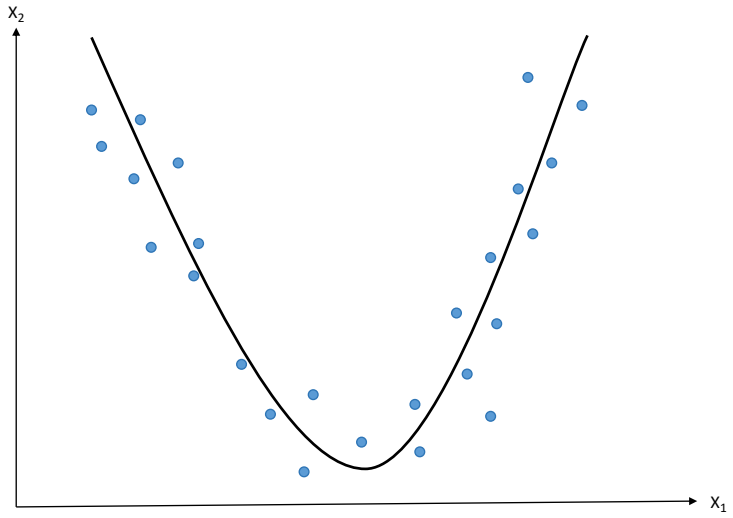


Figure 2.3: Regression illustration.

Figure 2.3 gives a visual example of regression. Predicting tomorrow's stock market value based on data from previous days or years is a classic example of a regression problem. In this example, tomorrow's stock market value is predicted and the resulting output is a continuous value [8, 9].

Classification

Classification problems are those where data is used to predict ordinal or nominal outputs. In these types of problems, the classification algorithm or function is used to split the data into multiple categorical classes. A similar process to regression can be used in classification with the caveat that the loss function in classification needs to be tailored to categorical prediction errors. The new loss function $L(k, j)$ will correspond to the penalty for classifying an observation from class k as j. Typically, a zero-one loss function is applied, where the value for an element being incorrectly classified is 1 while a correct classification corresponds to a 0. The classification model error is then the sum of all values

$$\text{Error} = \sum^{\text{for all } y} L(y, \hat{y}) \tag{2.4}$$

A well known example of a classification problem is the handwritten digit classification. In this example the images of handwritten digits are used as inputs and the algorithm is tasked with labeling each image with a value 0 to 9 [10].

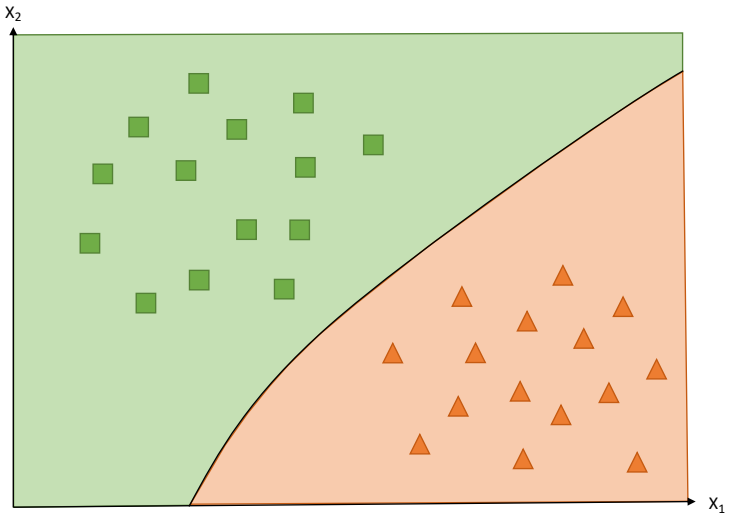


Figure 2.4: Simple classification illustration.

A simple binary classifier is visualized in Figure 2.4 where the two dimensional space is

carved into two sections, any new observation would be classified according to the section in which it falls. This visualizes the difference from regression which attempts to find a function that follows the data's trend and classification which finds the best way to separate data by carving the feature space into sections. Because our problem of network traffic classification is a supervised learning classification problem, our focus is on the models available for classification.

2.2.2 Classification Models

Regardless of the problem type, all supervised learning problems need to be trained on the available labeled data in order to enable the model to make predictions on new unlabeled data. The labeled data is split into training and testing subsets. This data split allows the use of part of the data to train or fit our model and another part of the data to test our model for model performance analysis. Once the data has been separated into train/test subsets, the classification model is selected.

It is of great importance to note that adjusting the model too closely to the training data may cause overfitting, where the model is so tightly matched to the training data that it will produce poor results when presented with new data. On the other hand, underfitting, where the underlying structure of the data is not discovered, can also provide poor classification results on unseen data. This is a delicate balance that must be accounted for by carefully selecting the parameters available to each particular model [10]. There are many classification models, because of the large number of classes possible in network traffic classification, the following are considered for this work:

KNN

k-Nearest Neighbors (KNN) is a non-parametric, lazy, distance based algorithm [9]. Non-parametric refers to the fact that KNN does not follow a predetermined structure, but rather the structure of the model is determined by the data. KNN is a lazy algorithm

because there is no computation in the training phase as training simply consists in saving the data points and all the distances for classification are conducted in the testing phase of the model. The new data point to be classified will be labeled according to the classification of the k closest neighbors. The more neighbors that are required to classify a new data point, the more time and memory intensive the process becomes. To predict a value's class, the distance metric from the new data point y to each of the training data points x is calculated. This is often measured using the Eucledian distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.5}$$

Where the distance between x and y is calculated, n being the number of rows in the training data set. While Euclidean distance is often the function selected to calculate distances, others like Manhattan, Minkowski or Hamming distances are also routinely applied. It is also common for the data in each feature to be standardized to zero mean and unit variance.

$$z_i = \frac{x_i - \bar{x}}{s} \tag{2.6}$$

Where z_i is the standardized value, x_i the value to be standardized, $\bar{x}$ is the feature mean and s the feature standard deviation. This is done to remove the bias introduced by large numerical values which can overpower smaller values when calculating distances.

Once the distances have been computed, the probability that the new data point y belongs to each class c from the k nearest points is calculated,

$$P(Y = c|X = x) = \frac{1}{k}\sum_{i=1}^{k} I(y_i = c) \tag{2.7}$$

Where the k nearest neighbors to $\hat{y}$ are analyzed and the function $I()$ produces a 1 when y_i is equal to class c and a 0 otherwise. This is will yield the probability of membership of $\hat{y}$ into all possible classes present in k neighbors, with the highest probability deemed the predicted label for $\hat{y}$ [9].

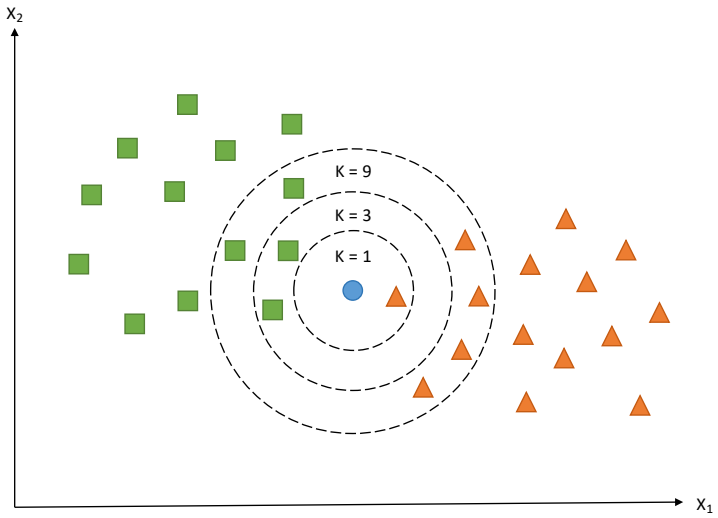


Figure 2.5: KNN illustration.

Figure 2.5 visualizes k-Nearest Neighbors. As the figure shows, the new data point marked as a circle will be classified as a triangle if $k = 1$, as a square if $k = 3$, and again as a triangle if $k = 9$. This shows the importance k has in the classification outcome using this model. It is important to note that the size of k has an immense impact on the model. If k is small, the model can suffer from overfitting, while a large k can produce underfitting.

Decision Tree

The decision tree algorithm, although simple to understand, is a powerful classification model. It has a tree structure, similar to a flowchart, where each node is a decision point based on a feature's value and at the end, or leaf, of the structure a class label is present. The training data is analyzed and partitioned based on a rule generated by the model. The two subsets are independently analyzed and partitioned again based on another rule

generated. This recursive partitioning is conducted until all the elements in the subset contain a single class label or until the splitting no longer adds any classification value. There are two main metrics to calculate the best rule to split any particular set of data: the gini impurity and the entropy or information gain. These are the criterion used to measure the quality of a partition. Although decision tree models can become computationally expensive to train, they do not require much computation to classify new data once trained. Additionally, decision trees are able to handle both categorical and continuous features [9, 10].

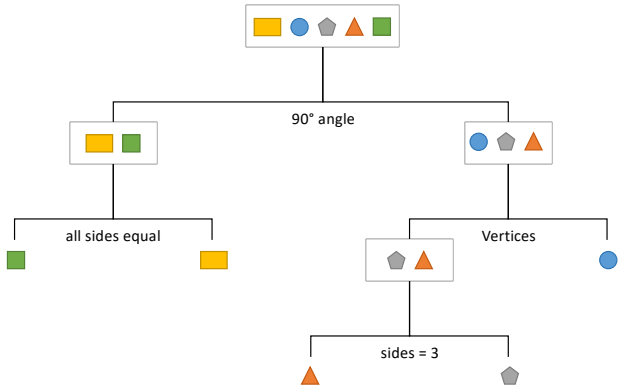


Figure 2.6: Decision Tree illustration.

Figure 2.6 visualizes a decision tree. All figures begin at the top of the image. At each intersection a rule on an a feature of that shape is tested. The result of the rule separates the shapes into smaller groups until there is only one possible shape at the bottom of the image, these are the leaves of the decision tree.

Random Forest

Because decision tree models recursively partition the training data until it is no longer advantageous, it is susceptible to overfitting. The random forest model addresses this concern by constructing several decision trees during training and merges the individual

decision tree's result for a more accurate and stable prediction. Random forest combines the decision tree model with bagging (bootstrap aggregation), where several subsets of the training data are selected with replacement. Each subset then randomly selects a subset of features, from which it begins generating a decision tree classifier which it grows to the largest possible size. Once all the subsets have created their respective decision trees, the random forest is ready to classify new data. The random forest model feeds the new data to all of the generated tree models and the random forest prediction is the aggregated prediction from the decision tree models [8, 9].

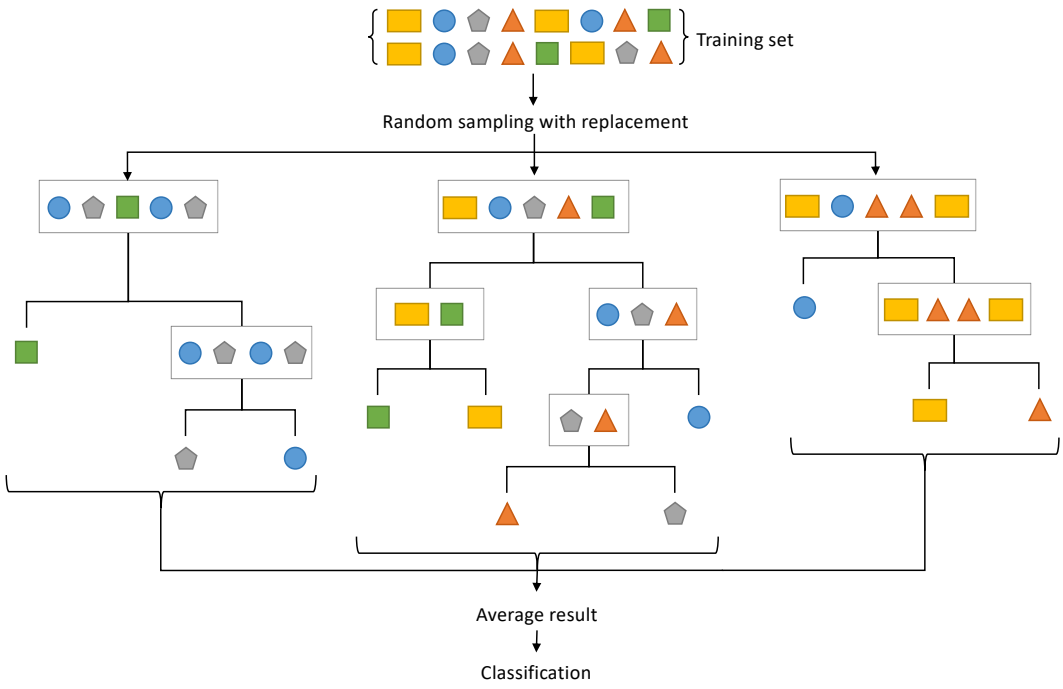


Figure 2.7: Random Forest illustration.

Figure 2.7 visualizes the random forest model. As is shown, several trees are generated from random samples of the training data. Although all the possible classes, in this case geometric shapes, possible are not present in every decision tree it is clear that once the results are averaged the resulting class is correct. For example, if a triangle is presented to

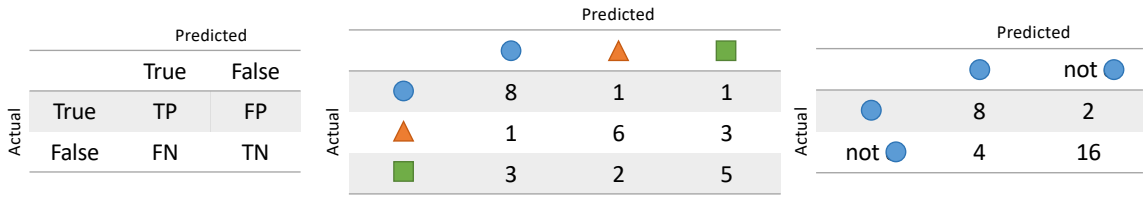
this random forest for classification, the result of the second and third trees will result in a triangle. Even though the first tree will classify incorrectly, the average will still classify as triangle. This shows how a random forest can produce a better classification model than a single tree. Similarly to decision trees, random forests can become computationally demanding during the training phase, but testing results are considerable faster since there is only a small calculation being done at that time.

2.2.3 Evaluation Metrics

A method for assessing performance regardless of machine learning algorithm is essential as it will allow comparison across algorithms. The confusion matrix is a great tool to evaluate classification performance. As you can see in the generalized form of the confusion matrix in Figure 2.8a, there are four values in the confusion matrix:

- True Positives (TP) - Marked as belonging to this class, correct.
- False Positives (FN) - Marked as belonging to this class, actually does not.
- True Negatives (TN) - Marked as not belonging to this class, correct.
- False Negatives (FN) - Marked as not belonging to this class, actually does.

Each row of the multi-class confusion matrix represents each of the class or group predicted while each column represents the actual values of each class. Figure 2.8b gives a visual demonstration of a confusion matrix. Here we can see how many circles, squares and triangles were classified as circles, squares and triangles. Figure 2.8c gives us the confusion matrix of the circles. Note that the False Positives are the sum of all the squares and triangles labeled circles, while the False Negatives are all the actual circles that are not labeled circles. Once the TP, TN, FP, FN values are obtained for each classification label, evaluation metrics can be obtained.



(a) General.

(b) 3×3 classification.

(c) Circle classification.

Figure 2.8: Confusion matrix examples.

Accuracy

The accuracy is the number of correctly classified over the total number of elements for a particular class or classification label,

$$Accuracy = \frac{TruePositive + TrueNegatives}{TruePositive + FalsePositive + TrueNegatives + FalseNegatives} \tag{2.8}$$

For multi-class problem, the overall accuracy is the sum of all True Positives for all classes over the total number of elements classified,

$$OverallAccuracy = \frac{\sum_{i=1}^{\text{all classes}} TP_i}{\#\text{ of elements}} \tag{2.9}$$

This shows how well this technique does at classifying the data. The overall accuracy is frequently utilized when comparing the same machine learning algorithm while tuning hyper-parameters [9, 11].

Precision

Precision is the percentage of elements that are adequately labeled out of the total number of elements labeled to a particular class. To calculate precision for any particular class:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.10}$$

This is a useful metric when a False Positive is a more costly error than a False Negative. For example, in spam detection, it is a greater penalty to mark an important email as spam (False Positive) than to let a spam email through (False Negative) [10, 11].

Recall

Recall is the ratio of elements which are correctly classified from the entire group of elements which truly correspond to that label. It tells what percentage of a particular class were able to be detected correctly. To obtain recall for any particular class:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.11}$$

This metric is especially important when doing any kind of health detection, as incorrectly classifying an ill patient as not sick is a very costly error [10, 11].

F1-score

F1-score is the harmonic mean of precision and recall. This allows for a single number to give a combined representation of the classification results.

$$\text{F1-score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{2.12}$$

The combination of precision and recall, often weighted by class element membership to avoid class imbalances in the metrics, are the two most telling metrics as it applies to our network traffic classification problem. For this reason, these will be the metrics utilized for the experimental analysis conducted as part of this work.

2.3 Network Application Classification

There are two methods to capture network traffic, *active* data capture where traffic is generated and injected into a network to perform measurements and analysis, and *passive*

data capture in which a measurement point is placed on the network and the typical traffic generated by users is observed. Active network traffic measurements are mainly utilized for fault and vulnerability detection or in some cases for testing application performance. For the purposes of network traffic classification, active measurements faces two main issues:

- Injecting network traffic alters network usage metrics. Network routers and switches must react to route newly created traffic, potentially altering network routing while reducing network throughput during active measurement data capture.
- Scalability becomes an issue. Large networks with many end systems require many experiments to test all possibilities. As the networks increase in scale, so to does the complexity for active measurements.

Passive measurements consist of observing traffic present in the network and collecting this traffic data for analysis. Passive data capture gives a greater insight into the network traffic as typical network behavior transmissions are observed and analyzed [6].

There are several approaches for network traffic classification using passive data capture on a network:

2.3.1 Using Port Number Conventions

Port numbers, along with the tranport layer protocol, are used to recognize which communication packets belong to each of the many applications running on that particular host. The Internet Assigned Numbers Authority (IANA) transport layer port numbers are classified based on three ranges: System Ports (0-1023), User Ports (1024-49151), and the Dynamic and/or Private Ports (49152-65535) [12]. IANA assigns application names to ports in the order in which they are registered.

To classify network traffic, the transport layer protocol and observed port numbers are read from the packet headers and compared to the IANA registered port numbers list to determine the application based on this mapping. The advantages of this approach

are that it is a simple procedure to classify network traffic. It also allows to incorporate new applications easily, as it is only a matter of appending the new application to its corresponding transport layer protocol and port number. Applications such as e-mail, FTP and DNS have been successfully classified using this port number approach [13].

There are two main obstacles for solely relying on port number conventions for network traffic classification. First, applications need to be registered in order to be detected. Unregistered or new applications that have not yet completed the registration process are therefore not supported. Second, there number of applications that are using other ports (e.g. using TCP/port 80 which is registered as HTTP for chat or streaming while avoiding firewalls) or dynamic ports (which are not registered by IANA) to obfuscate their traffic are increasing [14, 11]. The inability for this port based approach to keep up with the growth and direction of network traffic impedes its use as a reliable approach on its own for network traffic classification.

Keeping with the post office analogy from Section 2.1.1, this would be similar to classifying the letters and packages based solely on the envelope or box which contains them. While many retailers have their logos on their boxes which can make them easy to identify, such as the Amazon logo or mail from a university where their logo and information is printed on the envelope, many will reuse the boxes where now the package does not correspond to an Amazon shipment. Additionally, many use generic envelopes or boxes with no differentiating markings, similarly to network traffic using HTTP port 80, which makes those extremely difficult to classify.

2.3.2 Using Packet Payload Data

Definition

In addition to utilizing the packet's headers, this approach inspects the packet's payload, the actual information contained in the transmitted packet. This is often referred to as deep packet inspection (DPI) and can be used to identify the application producing this

traffic. The computational burden of analyzing every packet of each user's network traffic makes this approach unscalable. This is circumvented by generating unique identifier byte sequences, called signatures, to identify traffic. For example '$\backslash GET$' is used to identify web traffic, while '$\backslash xe3 \backslash x38$' can be found in P2P network traffic [13].

There are several issues with implementing this approach for network traffic classification. First, this approach is looking into the packets' payload which can create the potential for a violation of privacy with some countries having gone as far as to forbid network managers from looking into packet payloads. This limits its use to networks where these countries operate, which includes any international network backbone. Second, many times this signature analysis is done off-line, meaning packets are first recorded and then the signature matching is done on the recorded packet traces at a later time. Because capturing the entire traffic of all users in a particular network requires excessive storage which makes this difficult, packet capture algorithms typically only store a subset of each packet (e.g. the first 200 bytes). This can lead to a situation where the packet payload is cutoff before the bytes match any signature and are therefore not classified [13, 11].

Applying the post office analogy once again, this is similar to scanning mail and envelopes with x-ray machines. Although the x-ray scan will not be able to see everything in the envelope or package, it can identify metal shapes to try to find potentially hazardous objects. Similarly, DPI can be applied to analyze a section of the payload in order to attempt to find packet signatures which would produce an improved classification of the packet.

In practice, payload approaches are used to establish ground truth to allow further experimentation using alternative methods.

Tool: nDPI

Deep packet inspection (DPI) refers to the process of analyzing packet payloads. This technique can be used while creating flow exports to increase network visibility. nDPI [15] is ntop's open source library for DPI. nDPI is an ntop extension of OpenDPI, another open

source project, since this has not been updated ntop continued this work while keeping this as an open source project. nDPI uses libpcap, an open source library, for packet capture.

In this program, an application protocol is defined by a unique numeric protocol Id, and its associated symbolic protocol name (e.g. Skype). In nDPI a protocol includes both network protocols such as SMTP or DNS, and application traffic over network protocols (e.g. Facebook and Twitter over HTTP). A protocol is typically detected by a traffic dissector written in C, but it can also be labeled by analyzing the packet's protocol/port, IP address, and protocol attributes. For instance the Dropbox traffic is identified by both the dissector for LAN-based communications, and by tagging as Dropbox the HTTP traffic on which the Host header field is set to *.dropbox.com. This allows to both detect known protocols on non-standard ports such as detect HTTP on ports other than 80, and conversely detect non HTTP traffic such as a Skype call on port 80 [16].

Each dissector is available in its separate C source file. These protocols have attributes such as default level 4 protocol (TCP, UDP, TCP/UDP) and port (80 for HTTP, 53 for DNS). This allows unclassified traffic to be passed by all possible dissectors in a "most likely first" manner. For example, if the unclassified packet is TCP port 80, it will first apply the HTTP protocol dissector. If it is not identified by the HTTP protocol dissector, it will move through the available TCP dissectors until it either finds a match or exhausts all possibilities.

Because Internet traffic is moving towards encrypted content often using SSL, nDPI includes a decoder for SSL/STL certificates to support encrypted connections. Protocols and subprotocols can be detected using the encryption certificate which allows identification of encrypted protocols (e.g. Apple iCloud) that otherwise would be undetected. Additionally, nDPI has the ability to support sub-protocols using string-based matching. This is because many new sub-protocols such as Apple iCloud/iMessage, WhatsApp and many others use HTTP(S) that can be detected by decoding the SSL certificate host or the HTTP "Host:" field. nDPI includes an efficient string-matching library based on the popular Aho-Corasick algorithm for matching hundred of thousand sub-strings efficiently (fast enough to sustain

10 Gbit traffic on commodity hardware).

nDPI will analyze the payload of up to the first 8 to 10 packets [16] to attempt to classify the type of traffic for each flow. This heuristic value has been recommended by ntop as there is little to no advantage in looking beyond for classification purposes while analyzing more packets for each flow can introduce performance degradation as the packets must be saved in memory in order to be analyzed.

Continuing with the post office analogy, nDPI would represent our x-ray machine. All postage is analyzed with this tool and marked according to its findings.

2.3.3 Using Flow Feature Data

Description

This approach classifies network traffic by generating network flows as opposed to classifying data at the packet level. There are several advantages when using network flows as opposed to regular packet capture to represent network traffic:

1. Widely deployed - They are integrated into routers, switches and firewalls among other packet forwarding devices. According to [6], a recent survey among both commercial and research network operators shows that 70% of participants have devices that support flow export.

2. Proven effective and reliable - Its use in security analysis, capacity planning, accounting and profiling, along with its use to comply with data retention laws demonstrates the confidence instilled in these export protocols to describe network traffic.

3. Significant data reduction - Multiple packets are aggregated into a single flow. This means that data reduction to the order of 1/2000th of the original volume can be achieved, which allows for historical storage of network traffic information in order to comply with rules and regulations for communication providers. For example, in communication providers in Europe must retain communication data for "the purpose

of the investigation, detection and prosecution of serious crime for a period between six months to a year" [6].

4. Sensitive to data privacy - The fact that only packet headers are used in flow export, it is typically less privacy sensitive than packet export.

It is worth noting that although packet aggregation into network flows are reducing the amount of data, the size of flow data, specially in high speed networks, can quickly exceed tens of terabytes. Therefore, this can be considered a form of "Big Data" and comes with all the challenges that this type of problem encompasses (e.g. exponential data growth, scalability, resource constraints).

In the post office analogy, using flow features is similar to using shipment information. Rather than looking at individual items, information can be clustered as network traffic data packets are aggregated into flows. Instead of using IP/port addresses consider aggregating mail by ZIP codes. There is a lot of value in understanding transit trends at the ZIP code level. A post office manager can prepare routes and allocate resources according to the ZIP codes that send or receive the most postage to ensure that all mail gets picked up and delivered on time. Note that this data aggregation can be done with the aid of computer software which can analyze all incoming mail and produce the aggregated data and display the results in a practical manner for the post office manager.

Tool: PMACCT

Pmacct [17] is a small set of open-source, multi-purpose passive network monitoring tools used to account, classify, aggregate, replicate and export network traffic as flow objects. These tools, refered to as daemons in pmacct, have two main functions: packet acquisition (*pmacctd* the NetFlow exporter) and packet processing (*nfacctd* the NetFlow collector). Both daemon types are based on the same overall structure with several modules. The main module is called the core process whose main focus is gathering packets from the network. The core is additionally responsible for filtering, pre-tagging and sampling. There

are one or more additional modules, called plugins, responsible for the packet processing. The core process is structured in a circular queue which is then further subdivided into multiple buffers. Once a buffer is full, the core will signal to the plugin to begin processing. Network data aggregation duties are shared amongst the core and plugins. The core is in charge of flow definition while the plugin handles accumulation of counters.

Core Process

The core process is separated into two parts: the upper which collects network data and the lower which handles plugin operations by aggregating, filtering and core communication to/from plugins. Because the way that data is collected for the two deamons is different, the developers decided on making a hard separation between the upper and lower parts of the core process. This allows the bottom part to serve as an abstraction layer between the upper core and the plugins. Additionally, it allows modularity in that if a new core is desired only the upper part needs to be developed. Finally, if a new plugin is created, it can be added seamlessly so long as it follows the hooking interface from the bottom part of the core to interact with all available cores.

The *pmacctd* daemon upper core obtains network traffic data by using the libpcap framework. Once a packet is received, pmacctd sets pointers to the starting point of protocol headers up to the transport layer (TCP/UDP). The network layer (IP) is reassembled to handle packet fragmentation, and this new structure is passed to the beginning of the lower part of the core.

The *nfacctd* daemon received data in the form of NetFlow/IPFIX packets sent from an exporting agent (NetFlow enabled equipment or probes). It only accepts data from the specific exporters with whom it is setup to communicate, which allows for multiple instances of the deamon to coexist and capture different NetFlow versions and/or subsets of traffic if desired. It is then dissected and the extracted information is sent to the lower part of the core.

The *abstraction layer*, or lower part of the core, uses the set of pointers setup in the

upper level and is responsible for data aggregation, applying any additional filtering and feeding the buffered data to the appropriate plugins [18].

Plugins

Plugins receive the full buffers containing the data from the core process. Then the plugin may conduct its process whether it may be saving the data to a flat file, to an SQL database, printing out to the console or creating a data exporter probe to forward this data to another service. For detailed information on a full list of currently enabled plugins as well as in-depth description of each plugin available, readers are encouraged to consult the CONFIG-KEYS documentation on pmacct's official github account [19].

nDPI with Pmacct

Including 'class' in the aggregation list setting in pmacct's configuration file enables the use of nDPI's library. This allows pmacct to obtain the data for IE 94, 95 and 96 (application Description, application Id and application Name) in the creation of IPFIX packets which provide valuable information for network traffic classification. Because of the privacy concerns discussed earlier, this is only done on a subset of traffic to serve as ground truth for machine learning classification experiment validation.

Continuing the post office analogy, pmacct is the computer software creating the shipment aggregation data that, when combined with our x-ray machine (nDPI), provides great insight into the use of the post office resources and can be a great aid in managing the postal service. The challenge is to now extract the knowledge from the data that these tools provide and apply them towards optimizing the management of the post office, or in this the network. While automating postal service shipping has been in practice for some time now, there is a great opportunity for network traffic automation to be improved.

2.4 Related Work

Now that the individual actors in network traffic classification have been presented, an analysis of past and current efforts in this complex problem can be investigated. The works are arranged according to the technique used for classification of network traffic. Figure 2.9 presents the taxonomy for this section while Table 2.1 contains a list of related works along with their associated algorithm for classifying network traffic, features used and applications or categories in which their data is classified.

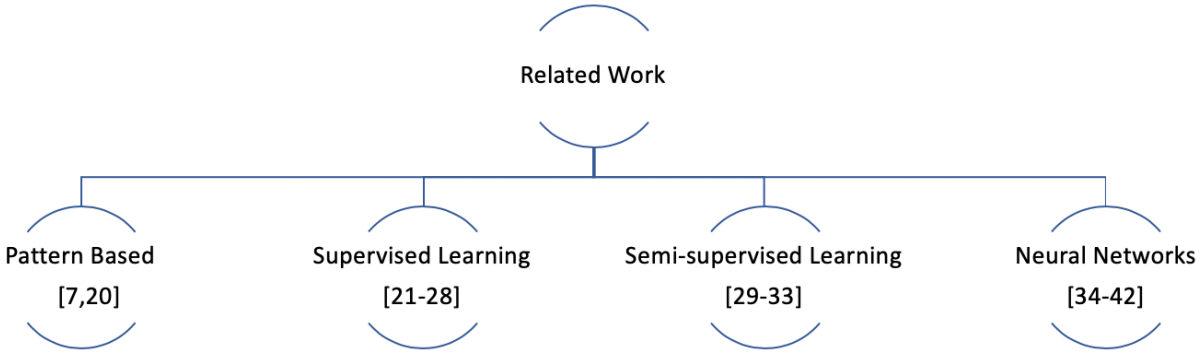


Figure 2.9: Taxonomy of related works.

2.4.1 Pattern Based Classification

BLINC

In their efforts to find a novel approach in classifying network traffic, Karagiannis et al. developed BLINC [7], a new approach which focuses on associating Internet hosts with applications based on behavior. In order to capture the host behavior, the authors evaluate flow data at three distinct levels. First at the social level, the interactions with other hosts are analyzed. They consider a host's "popularity" as the number of unique destination hosts with which a particular source host communicates. After analyzing these interactions the source and destination IPs are grouped into communities, by identifying and grouping hosts that interact with each other. The argument for doing this is that a

group of hosts can participate in a collaborative manner, or offer a service to the same set of hosts, such as a set of servers with different IP addresses belonging to the same service provider. Next, the number of source ports that a particular host uses for communication is explored. The argument for this analysis is that by analyzing the functional level of the hosts, one can further refine the type of service that a community is utilizing. Finally, the two previous levels are combined with more specific metrics such as transport layer protocol or average packet size in order to classify the network traffic behavior. These three levels are visualized into graphlets. Then, these relationships in the graphlets are used as visual signatures to classify all the traffic contained within that particular graphlet according to its behavioral visualization as pictured in Figure 2.10. This application level evaluation allows the combination of behavioral and general flow metrics into behaviors such as web traffic, online gameplay, P2P and streaming. BLINC provides a creative approach for network traffic classification, although it does have its limitations. A graphlet needs to be defined for each application in order to be able to classify that particular application. Also, if the case where there are similar graphlets exists, graphlets will no longer discern the applications effectively. The ideas behind this model for classification are worthwhile and a numerical representation can be created, which can be used along with data analytics to classify network application traffic.

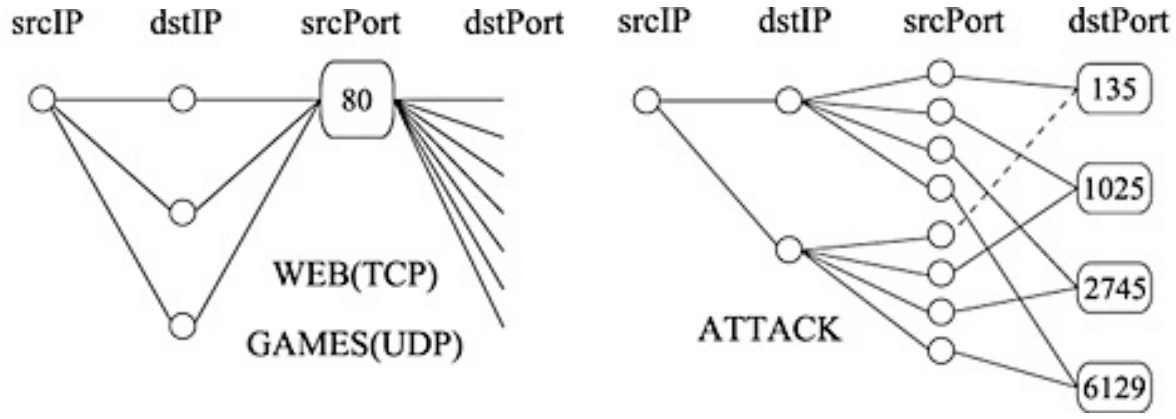


Figure 2.10: BLINC graphlet examples [7].

Traffic classification based on zero-length packets

Kampeas et. al. present a very interesting feature extraction approach for classification of network data. In [20], they propose that by considering communication at the application layer, a clear communication pattern is visible. Observing network traffic at the network layer, as the majority of researchers are doing, fragments this natural communication pattern as the data gets split into packets for transmission. The Application Protocol Data Units (APDUs) are the messages being communicated at the application layer. The back and forth communication of the APDUs exchange patterns are the focus of this work. They argue that although these exchange patterns, or signatures, are not unique as one application can have several signatures, the application can be identified with a high level of confidence with these patterns. In order to obtain these patterns, they develop a simple sequence named accumulated-APDU (a-APDU). A tuple is created representing the number of bytes in each direction, with only one side updating at a time, and at every switch in the direction of the data, a new tuple entry is added to the list of tuples representing this particular flow. Consider two sides (A and B), side A begins the communication and sends a message with 100 bytes. This creates a list where the index is the 4-tuple corresponding to the IP and port numbers of both A and B. The first tuple entry is created (100,0). Then B responds with 50 bytes. Since there is a change in the direction of the traffic, a new tuple is created (100,50). The list containing these patterns now contains two entries [(100,0),(100,50)]. Now, another message from B with 200 bytes is observed. Because there is no change in the direction of the traffic, the data is aggregated to the latest tuple, with the resulting list [(100,0),(100,250)]. These lists of tuples are what generate the patterns, or fingerprints, of the flows which are then used to classify network flows into applications. Because all that concerns them in this approach is the number of bytes each flow is sending, the authors develop a clever idea. Instead of looking at all the packets, they focus on the zero length packets. Because they focus on TCP traffic, the seq# and ack# in the packet header give all the necessary information to populate the signature lists. This means that they only need to capture and analyze a fraction of the traffic. On top of that, the calcula-

tion is a simple reading of a value in the packet header and the subsequent update to a list. This approach is therefore very amicable with online classification. Additionally, because they are not bothered by the payload, encrypted data is not an issue for this approach. This is a feature extraction approach, and any supervised classifier can be used to classify the signatures. However, the authors decide on the J48 decision tree classifier because of its wide use and ease in implementation for their experimentation. Once their data extraction approach is developed, they analyze a data set with 50 applications and a variable a-APDU signature length. The full list of applications is noted in Table 2.1. Their results show great outcomes for some applications. However, the authors note that 19 applications were completely missed. The authors note that these missed applications, such as POP3 and VNC, produce very poor performance because they only create 2 sequences. The authors also note that some applications see their evaluation metrics dip after the third sequence. The authors also compare this approach with a network layer approach where packet size is used to generate features. The results show a 3 sequence a-APDU outperforms this packet size approach as well as a flow statistics approach. This is an interesting feature extraction approach that, although it has its limitations as it does not work well with applications having few packets, can be combined with other approaches, such as neural networks or ensemble learning models, to take advantage of its resilience to encrypted traffic.

2.4.2 Supervised Learning

Machine learning in software defined networks: Data collection and traffic classification

The authors of [21] create a simplified framework to analyze and classify network traffic in an enterprise network. Their proposed framework consists of mirroring traffic from a switch onto an SDN enabled switch. This SDN switch acts as a filter, sending TCP traffic from a specified host through to the controller and dropping all other traffic. The specified host is sending controlled traffic generated by the researchers which allows for ground truth labels

to be placed on the generated traffic. BitTorrent, Dropbox, Facebook, HTTP, LinkedIn, Skype, Vimeo and YouTube are the applications for which traffic is generated and labeled. Once the data is captured and labeled, it is normalized and a Principal Component Analysis (PCA) algorithm is run on the data set to remove correlated features and only utilize linearly uncorrelated features in the experimental classification. Although the complete list of features captured is reported in Table 2.1, the uncorrelated sublist of features used for classification is not reported. Amaral et. al. apply machine learning classifiers to analyze the data. The classifiers utilized are Random Forests, Stochastic Gradient Boosting and Extreme Gradient Boosting. These classifiers are selected because they work similarly in that they use a set of regular classifiers and classify data by the weighted average of the individual predictions. To create their train and test split, a random number generator calculates n values which are used as the train set, with the remaining entries used as the test set. The models are then fitted with the trained set and accuracy is evaluated with the test set. This process is executed 30 times and the average of all executions is presented as the experiment accuracy score for each classifier. The reported accuracy of the classifiers has only a small variation within each of the applications classified, with the exception of YouTube and Facebook where the difference among classifiers is more substantial.

SVM-based Classification Mechanism and Its Application in SDN Networks

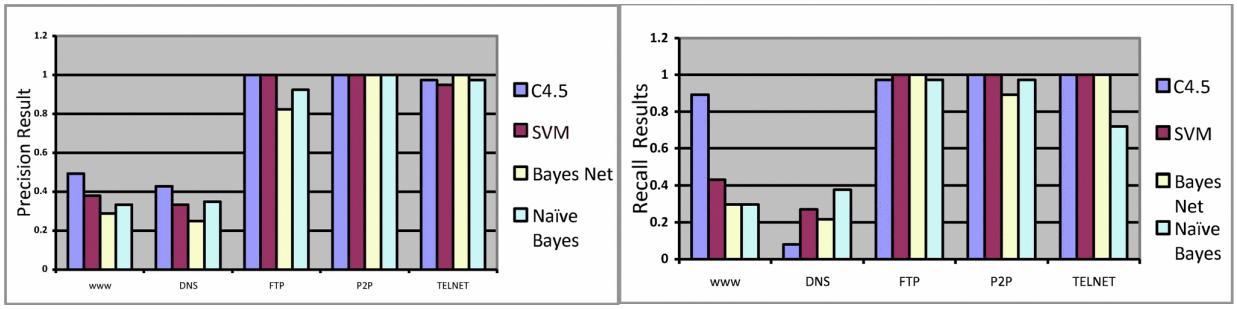
Liu et. al. present their work [22], where a mechanism for network traffic classification applied to SDN is developed. This is done using a network virtualization function (NVF) with the aim of applying network traffic classification to virtual networks, where the networking devices consists of software running on a cloud server. The support vector machine based Internet traffic identification and classification (STIC) mechanism consists of a virtual classifier communicating to a network controller. Data communicating through this virtual network is mirrored to the SVM classifier, where it is analyzed and a classification label is predicted. The network controller uses OpenFlow, a networking protocol that allows access to the data plane in a network device, to append a VLAN ID to incoming

traffic according to the classifiers results. This VLAN ID is used by the Open vSwitches to reroute the traffic according to the classification result. This allows for network traffic classification to influence routing within a network. The SVM classifier is tested with 4 data sets created and collected by the authors. The list of 29 applications represented in the data sets are shown in Table 2.1. Although there is mention of using a radial basis function (RBF) kernel for feature selection, which is not uncommon when using SVM, there is no mention of the features considered or selected. The overall accuracy, precision, recall and F1-score is reported with all between 84-88%. They also implement a decision tree in addition to STIC for the specific purpose of classifying YouTube traffic according to a combination of quality (144, 360, 1080 pixels) and time (4 and 20 mins). The results of this YouTube data show that using the features for both length and quality in a decision tree outperform the decision tree where only one of the features is used in isolation thus demonstrating value in both features for YouTube detection.

Network traffic classification techniques and comparative analysis using machine learning algorithms

In [23], the authors evaluate machine learning techniques in network traffic classification. Packet captures are made using Wireshark to capture WWW, DNS, FTP, P2P and Telnet traffic and then leverage the Netmate tool for feature extraction on captured packets. A total of 23 features are reported to be extracted from the captured data but the list of features is not provided. The data is then split into training and testing sets which are used to train and evaluate the machine learning models. Shafiq et. al. selected to test the C4.5 (Decision Tree), Support Vector Machine (SVM), BayesNet and NaiveBayes classifiers in their comparative analysis. Accuracy, precision and recall are provided for the resulting classification scores of each classifier and it is clear that the C4.5 classifier outperforms the rest. Figure 2.11 notes that this is largely due to C4.5 having the ability to recall the WWW traffic by over 80% while the other classifiers could only perform a maximum of 40% recall on that particular type of traffic. Additionally, DNS is poorly classified by all

classifiers.



(a) Precision (b) Recall

Figure 2.11: Machine Learning classifiers metrics by application [23].

A Comparative Performance Analysis on Network Traffic Classification using Supervised Learning Algorithms

Archanaa et. al. perform a comparative analysis of supervised learning algorithms in [24]. A data set from the University of Queen Mary's repository is used. This data set contains 266 features. Feature selection methods are applied in order to reduce the data to a more managable subset of features. PCA, CfssubsetEval, chi-squared and InfoGain are applied to the complete data set. CfssubsetEval with Best First search provides the smallest working subset consisting of 8 features:

- first quartile inter-arrival time
- Frequency of zero receive window advertisement
- Categorical value Y or N for SYN permission for SACK
- Total time for data transfer

- Number of RTT samples found
- Median Ethernet frame bytes
- Minimum arrival time between packets

The Naive Bayes, BayesNet and Complement Naive Bayse base classifiers are analyzed and the best performing of these base classifiers is BayesNet. Similarly, several ensemble classifiers implemented in Java's Weka package for data analysis and predicting modeling. Decorate, Random Forest, AdaBoost, Bagging, Stacking and Rotation Forest, are compared. The reported results show that Decorate provides the best classification precision and recall of the selected data set at 99.6% for both, with Random Forest a close second with 99.4% and 99.5% respectively. Comparative performance analysis are a good approach to evaluate the performance of various classifiers under the same conditions. This should be explored further with larger and more recent data sets.

Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification

The authors of [25] aim to combine applications based on the type of service they provide in order to classify and eventually prioritize network traffic to provide the required Quality of Service for the class of application in use. Four main classes are defined:

- Interactive (Telnet)
- Bulk data transfer (FTP-data, Kazaa)
- Streaming (RealMedia)
- Transactional (DNS, HTTPS)

These classes are analyzed using K Nearest Neighbor (KNN) and Linear Discriminant Analysis (LDA). There are four data sets used for these experiments, one comes from the

Waikato Applied Network Dynamics (WAND) group at the University of Waikato (New Zealand), and three Gigascope data sets from an access network on a T3 line using a Gigascope probe collecting TCP traffic. The features selected for classification are the average packet size, flow duration, bytes per flow, and root mean squared packet size. Roughan et. al. note that the two most valuable features for classification are the average packet size and flow duration. There are several interesting findings in this article. First, they conclude that many streaming applications act very similarly to bulk-data and thus these simple statistics are not ideal for separating the two. While examining these two more closely, they noticed that streaming traffic, while having a fairly consistent behavior, in many cases ended with a long gap (20-40 seconds) followed by a few (2-7) final packets as shown in Figure 2.12. They note that this is a protocol related effect, and that their statistical metrics are being biased by this effect. They opted to remove the last 10 packets from each flow. This eliminates this bias and allows for better separation of streaming from bulk-data transfer traffic. They also note that inter-arrival variability, which is defined as: $E[r] = \frac{1}{N}\sum_{n=1}^{N} r_i$ where $r_i = \sigma_i/\mu_i$ and N is the number of flows with at least three packets, also appears to provide a good metric to separate streaming from bulk-data transfer using LDA. Finally, they note that they experience consistently positive performance using 3 and 5 Nearest Neighbors throughout their work.

Toward Classifying Unknown Application Traffic

In [26] K Nearest Neighbors is used to classify network traffic. The difference is that Baker et. al. use the two-sample Kolmogorov-Smirnov (KS) distance, as opposed to the typical Eucledian distance, to calculate the nearest neighbor. The features used for their experiments are:

- Packet count
- Byte count
- Average packet inter-arrival time

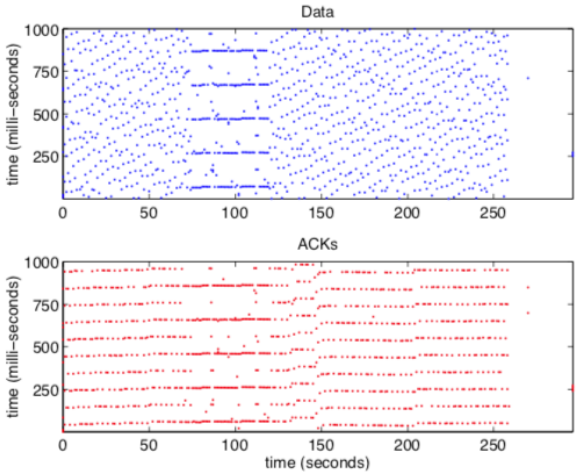
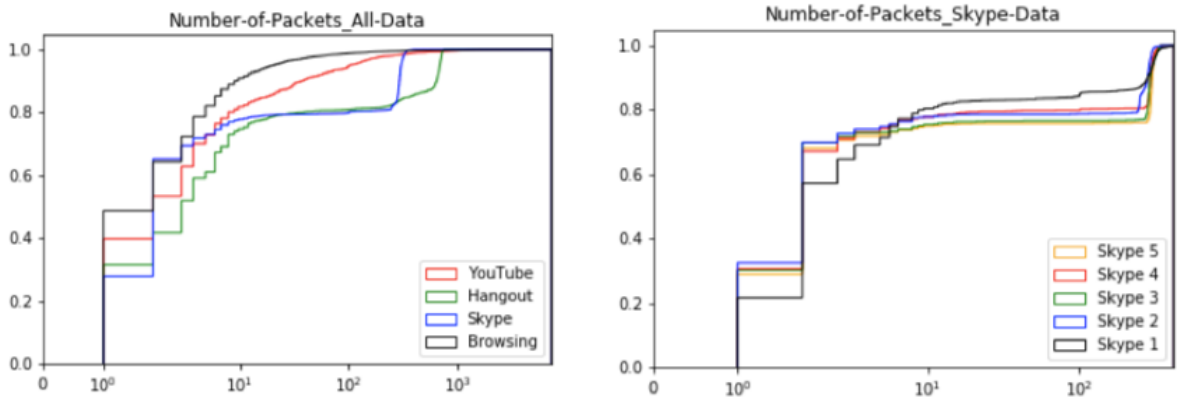


Figure 2.12: Streaming traffic shown in seconds on the x-axis and the milliseconds within that second on the y-axis. Note the trailing packets at the end of the flow [25].

- Average bit rate of connection
- Largest/Smallest packet size
- Longest/Shortest packet inter-arrival time
- Direction (inbound/outbound)
- Number of ARP packets
- Number of DNS packets
- Number of TCP ACKs
- Min/Max advertised receive window

The data used to evaluate this machine learning algorithm is captured using TCPdump. The desired traffic for evaluation came from Skype, Google Hangout, Youtube and HTTP web browsing. Figure 2.13a shows the cumulative distribution function (CDF) for each application evaluated which is the basis for calculating the KS distance. Figure 2.13b shows several CDF's of Skype which allows to perceive that the CDF for each application

is very similar while having a distinct CDF across applications. The use of KS distance is compared against the typical Eucledian distance and the results for these applications show a large improvement in classification. Specifically, while both present similar values for true positives in classification, the Eucledian distance yields 81.07% false positives while the KS distance only presents 7.19%. It is worth noting that this is done in a controlled data set and further testing is necessary to prove its validity when a diverse group of applications is classified.



(a) CDF of all applications

(b) CDF of multiple traces of Skype traffic

Figure 2.13: Cumulative Density Functions used for KS distance for K Nearest Neighbor classification [26].

Early classification of residential networks traffic using c5.0 machine learning algorithm

In [27] a framework is created to classify residential network traffic. The proposed solution relies on a supervised machine learning method applied to a set of statistical characteristics for the first packets of a flow for classification. nDPI is used as their ground truth as they note that DPI comparative studies have showed that only nDPI and Libprotoident are the two libraries with reliable accuracy. C5.0 decision tree is the successor of C4.5 and is the selected supervised learning classifier used in this work. The list of features used for

classification are listed on Table 2.1. Since this classification framework is using only the first few (5 to 10) packets for classification, Aouini et. al. remove any TCP bidirectional flows that do not contain any SYN flags, any UDP flows obseverd during the first 120 seconds of the capture and any flows where no ground truth is able to be determined in order to obtain a high quality data set. The reported accuracy is above 98% for the C5.0 classifier which easily outperforms the Naive Bayes, KNN and C4.5 classifiers with which it is compared.

Network traffic classification based on transfer learning

Sun et. al. introduce the concept of transfer learning into machine learning for network traffic classification. In [28] TrAdaBoost, a modified version of AdaBoost to enable transfer learning, is used to classify network traffic traces. Maxnet is used as the base classifier which is then enhanced with TrAdaBoost to allow for transfer learning. WWW, mail, database, FTP, P2P and Services are the applications used for training and testing this approach. The number of flows in the WWW and mail are reduced as using all of them would result in an imbalanced data set. The selected features to be used for classification are shown in Table 2.1. Their experimental analysis consists of two conventional versions of MaxNet and the transfer learning TrAdaBoost. Their results show that using TrAdaBoost with 5 MaxNet base classifiers reporting a 98.7% accuracy, where the traditional MaxNet classifiers performed at 81.2% and 85.45%.

2.4.3 Semi-supervised Learning

PSO optimized semi-supervised network traffic classification strategy

This work [29] improves on the K Nearest Neighbor algorithm by applying both a semi-supervised learning approach to reduce the number of required training samples as well as Particle Swarm Optimization (PSO) in order to reduce KNN prediction time to enable its use in real-time network traffic classification. The K-means clustering technique,

an unsupervised technique where the algorithm groups data into k clusters based on the characteristics of the data, is applied on the training data containing some samples that possess application labels. Once clustering has been conducted, the training labels are used to assign an application label to the cluster to which it belongs. This now fully-labeled data set can be used as training data for KNN. PSO is an optimization technique where a population of individuals, or neighbors in the case of KNN, can be considered at once. This optimizes KNN in that instead of needing to find the distance measure to the closest k neighbors one at a time for any prediction, it can find the closest neighbor by considering all neighbors at once. The resulting algorithm is named PSO-KNN and is compared to C4.5 (a decision tree algorithm), NBK (Naive Bayes with kernel density estimator), K-means clustering, and KMKNN (K nearest neighbor with k-means clustering acceleration) to verify its ability to correctly classify network traffic. The reported results on classifying WWW, FTP, RTX, bulk data transfer, and RTMP traffic show that C4, KMKNN and PSO-KNN have the best results with similar accuracy scores. Although the author states that the PSO-KNN reduces the computational complexity of KNN, which seems reasonable since computations are done simulaneously as opposed to sequentially when calculating the nearest neighbors, there is no metric or analysis to support this claim. This is an interesting approach to optimize the prediction time using KNN. However, further analysis needs to be conducted with a large data set to confirm scalability as well as quantification of the time complexity reduction.

A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs

The authors of [30] approach network classification from the angle of its applicability in Software Defined Networks (SDN). Wang et. al. focus their efforts in traffic classification for the purpose of providing adequate quality of service to the end user. Because their focus is not on the specific application (i.e. Skype, Google Hangout, etc) but rather on the coarse-grained behavior (VoIP) the broad classification classes are:

- Voice/Video Conferencing (Skype, QQ, Google Hangout, etc)
- Streaming (PPStream, Vimeo, SopCast, Putlocker, etc)
- Interactive Data (Gaming, Web, HTTP Services, etc)
- Bulk Data Transfer (FTP, Torrent, Dropbox, etc)

Their framework consists of two stages, the first stage detects elephant flows from incoming traffic while the second stage classifies detected elephant flows. Keeping with their aim of adequate quality of service, they argue that only elephant flows are of interest to satisfy their needs. They characterize "elephant flows" as those flows that use above a threshold of a particular link's bandwidth, ranging from 1% to 10% depending on the actual bandwidth of the link. For their Machine Learning algorithm the authors selected to use a semi-supervised Laplacian Support Vector Machine (SVM). They opt for a semi-supervised machine learning model based on the idea that because all applications grouped in the same classification class require the same quality of service, their tendency is to exhibit similar statistical properties. This idea also allows unlabeled flows to be classified with the trained model and then subsequently use these newly labeled models to train the algorithm. In order to evaluate their framework the model is trained and tested using a 59GB traffic trace file captured by the Broadband Communications Research Group in Barcelona, Spain. This data set contains over 760,000 flows of which 440,000 torrent flows are removed as the authors argue that it would have lead to a highly imbalanced data set. 3,377 of the remaining flows are considered elephant flows which are labeled using DPI. The flows are then split into training and testing groups using a ratio of 7.82 training flows for each testing flow. From the 60 available features extracted for each elephant flow a subset of 9 is selected as training a model over a larger subset no longer provides any benefit as can be seen in Figure 2.14. Additionally, utilizing such a wide range of features without enough flow samples would lead to a severe overfitting of the classifier model. The most reliable features for the model are selected using a feature selection algorithm (Wrapper method)

in which forward selection is employed. After executing the feature selection algorithm it is clear that there is no improvement on the model by using more than 9 features for this particular problem. Wang et. al. report classification of elephant flows using their framework with 90% accuracy.

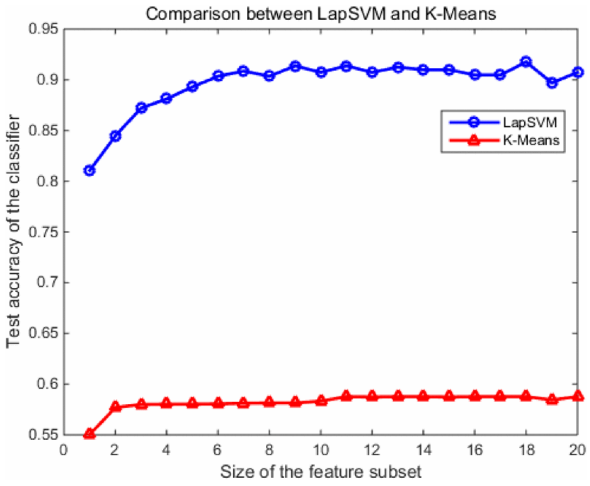


Figure 2.14: Classifier accuracy relative to the number of features selected [30].

QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs

[31] provides a framework for quality of service aware flow classification. Yu et. al. note that because network traffic is constantly changing, classification must be able to adapt with the traffic. They argue that semi-supervised is best for real time applications as DPI cannot recognize all of the flows with the increasing amount of encrypted traffic and this leaves only a subsection of labeled data. At the center of their framework is a tri-training semi-supervised learning mechanism. They select a Support Vector Machine, Bayes classifier and K Nearest Neighbor classifier for their framework. The 4 classes used to classify are voice, video, bulk data and interactive data. The training data, for which only elephant flows are selected, is split amongst the three classifiers. Once trained, the weights assigned to the different classifiers in this framework are determined by the correctness of the classification

during the training process. After conducting the experimental plan, they report to having found 8 to be the most appropriate number of flow features. The results show nearly an 11% accuracy improvement over the traditional tri-training.

A Framework & System for Classification of Encrypted Network Traffic using Machine Learning

Seddigh et. al. build a Machine Learning Traffic Analysis Tool (MLTAT) in their work [32]. This framework aims to train, tune and validate machine learning models for network traffic classification whose predictions are combined to give a more robust classification. Logistic regression, support vector machines, decision trees, adaboost, neural networks and naive bayes are all combined using bagging. There are two specific forms of bagging implemented, a majority vote and a weighted vote based on each individual classifier's confidence. MLTAT is trained using the following bidirectional network flow features:

- Min, max, mean and variance of packet inter-arrival time in both directions
- Min, max, mean and variance of packet size in both directions
- Total flow duration
- Protocol
- Total packet, byte and payload count in both directions
- Entropy of packet size
- Inter-arrival time in the "backward" direction

Additionally, they develop a two phase semi-supervised data capture technique to label data for training and model evaluation. First, type 1 data refers to fully labeled data that is acquired by generating and capturing a particular application's network traffic. Because this traffic is being generated by the authors, they can be certain of the ground truth label

for that traffic. It is worth noting that the authors express their difficulty in setting up a machine that would only communicate the desired application's message, as many system and network advertisement and signaling packets for network operation are routinely sent and inevitably became part of the data capture. Once type 1 data is labeled, type 2 data is collected from a university network. This unlabeled traffic data capture containing many applications is then labeled using a co-training approach based on another author's work. This entails generating two separate random forest classifiers using independent features. One focuses on packet related features while the other on time related features. The generation of training data is a three step process.

1. The type 1 data collected is used to train both random forest classifiers. These are then used to predict on the unlabeled data. Those flows with a matching prediction across both random forest classifiers and a confidence level greater than 80% are labeled and incorporated into the new training set along with the type 1 data.

2. The new training set is used to train both random forest classifiers once again, with the remaining unlabeled data classified with these newly fitted models. This time, if either of the classifiers has a confidence greater than 80%, that flow is labeled and added to the training data set.

3. In the final phase, three classifiers (random forest, neural network and adaboost) are trained with the new aggregated training data. This time, all remaining unlabeled data is classified by a majority vote of the three models.

This is a very interesting approach to generating training data for network traffic classification. The authors report that this approach provides over 93%, 91% and 88% accuracy when 20%, 10% and 5% respectively of the initial data is labeled. It is also reported to have outperformed decision tree and KNN classifiers.

A New Semi-supervised Method for Network Traffic Classification Based on X-means Clustering and Label Propagation

Noorbehbahani et. al. present a semi-supervised approach for generating training data to apply machine learning algorithms for network traffic classification [33]. They apply x-means clustering, a clustering algorithm based on k-means clustering. K-means clustering creates k clusters from the unlabeled data. X-means builds on that by constantly attempting to split each cluster until a criterion is reached. In this case the Bayesian Information Criterion (BIC) is selected. BIC is the optimization function that x-means is optimizing by finding the number of clusters where BIC is lowest. Once the unlabeled data has been clustered, the subset of labeled data is added and the unlabeled data is classified according to the k closest points within their assigned cluster. The authors opt to use the publicly available Moore data sets. The classification classes contained in the data sets are bulk data, database, interactive, mail, services, WWW, P2P, attack, games and multimedia. The applications contained in each class can be found in Table 2.1. The experimental plan consists of using 20% labeled data using 3 or 5 nearest neighbors on 3 distinct data sets, corresponding to 6 experiments. Once the data has been labeled according to this approach, J48 (decision tree) and naive bayes classifiers are created using the training data to compare the accuracy in prediction using the original labeled data against data labeled according to the author's approach. The results show that both models perform within 1% of each other in all cases. This is an interesting approach to generating training data which has been proven to work on the Moore data sets which are known for having high variance and thus are a good measure of this approach's ability to generate labels for training data.

2.4.4 Neural Networks

Packet-based Network Traffic Classification Using Deep Learning

Lim et. al. present an interesting approach for network traffic classification in [34], where packet payload data is converted into images and then classified using a convolutional

neural network (CNN) as well as a residual network (ResNet). The packet payload is transformed into an image by considering each byte a pixel on an image. There are 4 pre-determined image sizes (6×6 pixels, 8×8 pixels, 16×16 pixels, 32×32 pixels). If the image contains less pixels than those generated from a packet's payload the surplus pixels are truncated. Conversely, if the image is larger than the number of pixels generated from the packet's payload, the remaining pixels are filled with trailing zeros. The selected applications for classification are Remote Desktop Protocol (RDP), Skype, SSH, BitTorrent, Facebook (HTTP), Google (HTTP), Wikipedia (HTTP) and Yahoo (HTTP). 10,000 packets randomly selected from each application's packet payload are used to train the networks. For the output data of the network, a one-hot encoding vector is used. This means that there is a 1×8 vector where all the entries are 0s and a single 1 is used to distinguish the label representing the application. The two deep learning models selected for this article are Convolutional Neural Network (CNN) and Residual Network (ResNet). The CNN contains the input layer, two separate convolution layers, a pooling layer and finally a fully connected output layer. The convolution layers, as the name implies, are layers where the image is convolved with itself. The ResNet consists of an input layer, an initial convolution layer, a convolution group layer that consists of a series of 3 convolutional layers, a second convolution group layer, a pooling layer and a fully interconnected output layer. The ResNet architecture is visualized in Figure 2.15. The key in ResNet

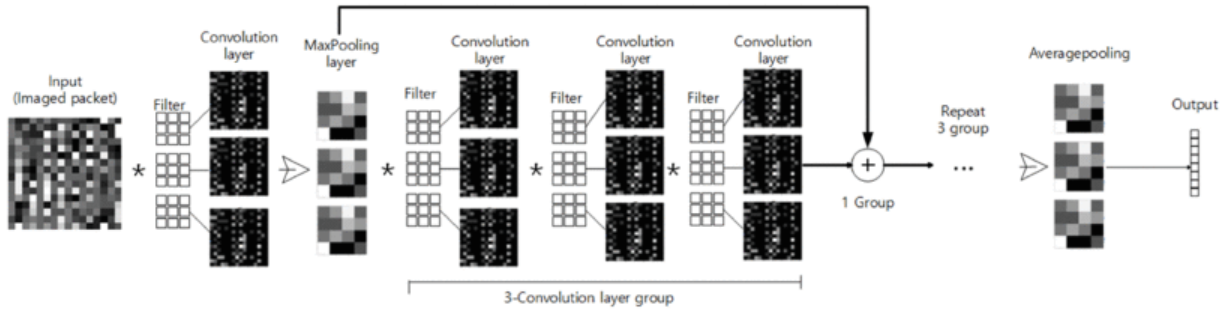


Figure 2.15: ResNet architecture [34].

is that there is a "shortcut connection" where the output of the initial layer is fed not

only to the first group convolution layer, but also added to the output of the third group convolution layer. This means that the input of the second group convolution layer is the sum of the input and output of the first group convolution layer. The F1-scores conclude that the CNN achieves better performance than ResNet when using smaller images (6×6 pixels) while ResNet outperforms CNN with larger images (16×16 pixels, 32×32 pixels) while in the intermediate size images (8×8 pixels) the performance is almost identical.

A Traffic Classification Method Based on Packet Transport Layer Payload by Ensemble Learning

Xu et. al. present a novel ensemble approach to network traffic classification in [35]. The authors select 1DCNN (1-dimensional convolutional neural network), 2DCNN (2-dimensional convolutional neural network), and LSTM (Long Short Term Memory) as the base models to which they apply a bagging strategy. Using a publicly available data set, the applications to classify are combined into 6 categories: email, chat, stream, P2P, VoIP and file transfer. Each packet from the data set is stripped of the header and only the first 784 bytes in the packet payload are used, filling shorter packets with trailing zeros. These bytes are converted to decimal values and arranged as both a sequence and a 28×28 matrix. This is done in order to prepare the data for the 1 and 2 dimensional CNNs. The data is split into 10 data subsets, 9 data sets to train and 1 dataset to test the trained models. Each of the 3 base models is assigned 3 training files and each generates 3 separate models by training each with a separate training file. This results in a total of 9 trained models. Once the models are trained, the bagging strategy is implemented in the prediction of classification labels. This corresponds to combining the results of the separate models to obtain a final result by selecting the label with the majority of models predicting it as the resulting class. The experimental analysis conducted provides positive results for most categories, with the exception of chat and mail. The authors attribute the subpar results for these two categories to the relatively small number of samples available in the data set which makes training more difficult. Finally, this ensemble learning model is compared

to C4.5 (decision tree model) and KNN experimental results conducted in another paper which used the same data set for training. The results of the comparative analysis show that the ensemble learning model presented here outperformed both C4.5 and KNN.

Byte Segment Neural Network for Network Traffic Classification

A byte segment deep neural network (BSNN) for classification of network traffic is presented in [36]. Their idea consists of using deep neural networks to classify networking packets by looking at the payload data and finding patterns in the datagram. The data packet is removed of all packet headers so that only the packet's payload remains. The payload is then separated into fixed-length segments that will serve as input for the BSNN. Each segment is transformed with a recurrent neural network which serves as an encoder to generate a representation of the data sequence. They select two different recurrent neural network (RNN) encoders, one using a Long Short-Term Memory (LSTM) and another with a Gater Recurrent Unit (GRU). Once the data is encoded, focal loss is involved in the BSNN to account for the imbalanced nature of network traffic data sets. The resulting encoded segments are analyzed by a multilayer perceptron which outputs classification with a softmax output layer activation function. For the model fine-tuning, a data capture is conducted containing data from 10 applications: DNS, BitTorrent, PPLive, QQ, SMTP, 360, Amazon, Yahoo, Couldmusic and foxmail. In order to find the optimal number of segments required they compare different values and conclude that 8 segments is sufficient for classification without inducing a timing penalty which could hinder their ability to apply this in real-time environments.

In order to compare BSNN to other packet based techniques, the authors select the deep packet inspection tool nDPI and Securitas, another datagram level traffic classification method which extracts features to then feed to the typical supervised learning algorithms. The authors decide on using Securitas with SVM, C4.5 decision tree and a Bayes neural network. Although their data capture contains 10 applications, their experimental analysis focuses on QQ, PPLive, DNS, 360 and BitTorrent. Also, because Securitas is a

binary classifier, a model for each of the supervised learning algorithm with each of the 5 applications is created. The results are presented, which show that BSNN with a LSTM encoder and Securitas-C4.5 produce the best results.

In a second phase of analysis the applications Amazon, Yahoo, Cloudmusic and foxmail are evaluated against BSNN-LSTM. Because these are considered novel applications, they provide insight of BSNN's ability to classify new applications correctly. Results for these novel applications are reported with all having a per application F1-score over 85%, showing the model's ability to classify new applications. Finally, Li et. al. report that the average time for processing a single datagram using BSNN is $2.97ms$. Securitas, which is currently used for real-time online classification, is clocked at $7.01ms$. This makes a strong case for BSNN ability to classify real-time network traffic. This is a very thorough article which provides an interesting alternative to the typical flow based network traffic classification.

Leveraging Inner-Connection of Message Sequence for Traffic Classification: A Deep Learning Approach

In [37], Jin et. al. develop a feature extraction approach by obtaining the message segment for the first 16 segments in a flow. A segment is defined as an independent piece of content transmitted between hosts. The extraction of a message segment from the TCP packets is done in the following steps:

1. Remove any packets consisting of only the acknowledgement or retransmission.
2. If an observed packet is of the maximum segment size (MSS), add this data to the current segment.
3. If an observed packet is smaller than the MSS, add this data to the current segment and then conclude the segment.
4. If a transmission is beginning in the opposite direction, the message segment ends.

Message segment sequences for applications show how applications interact and thus can be used to classify network traffic. The authors select a Long Short Term Model (LSTM), a recurrent neural network capable of "remembering" for a long time. In order to train the model a 3-day capture is conducted and only full TCP flows of HTTP, SSH, SMTP, WeChat and Remote Desktop are saved. In order to capture P2P and Video traffic, the authors recruit volunteers to watch videos or download files for five days and the traffic generated is captured. Model analysis is conducted using the captured data to determine the best hyper-parameters for this neural network. The results conclude that a 2 layer LSTM with a size of 400 neurons per layer yields the best performance. An experimental analysis is performed and the results report an accuracy $> 90\%$ for all applications including a perfect score for both Video and P2P.

Common Knowledge Based Transfer Learning for Traffic Classification

In [38], Xiao et. al. present a very interesting approach to preserve knowledge in a deep neural network (DNN). The purpose of this knowledge transfer is to allow the initial training of the neural network which performs the source, or initial, task to transfer a bulk of its knowledge to be applied to a subsequent target task. Their multi-output DNN approach separates the neural network into two sections. The input layer feeds into the common layers which store the bulk of the knowledge. These common layers then feed into private branches, a collection of networks that are trained independently from one another and are thus tailored to a specific task. The source task applies its data to train both the common layers and the initial private branch, then the second task uses the trained common layer in addition to its training data to only train the new private branch. This is done for as many tasks as needed and the result is a common layer with multiple independent private branches. The fact that only a private branch needs to be trained for a new task greatly reduces the training time complexity relative to training the entire deep neural network.

To validate their approach, 16 features from the WITS ISPDSL-1 and ISPDSL-2 data sets along with Moore's Set09 and Set10 data sets are used to train and test their model.

Because they are working on transfer learning, they select flow duration, flow rate and application classification as the 3 target outputs for each training and testing experimental sets. This means that they create a different model for each data set and each target output for all the models. Their model is compared by creating and training KNN, SVM and RF as well as a DNN without the multiple output option and their results are presented. Their experiments show that for all data sets, DNN and multi-output DNN are always the top 2 performers and always within 1% of each other. They then go on to compare DNN with multi-output DNN by presenting the perplexity of each. Perplexity is the normalized distance to the geometrical center of a label. The lower the perplexity value for any label, the better that the traffic is represented in that knowledge space. The perplexity score of multi-output DNN outperforms the traditional DNN in every target label. This demonstrates that although they produced nearly identical prediction accuracy scores, multi-output DNN has a more well defined separation of the output targets. This is a very interesting approach at reducing the time to train a model as well as give the ability to share knowledge which can become very valuable for future work.

An improved stacked auto-encoder for network traffic flow classification

The work in [39] leverages unsupervised learning and neural networks to create an improved auto-encoder for network traffic classification. Li et. al. improve on a traditional auto-encoder by training the network based on Bayesian probability theory. This is done by adding a softmax function to the output layer of the neural network where Bayesian theory is applied to find the weight values that maximize the likelihood that each training sample is assigned with the correct class label. They utilize two data sets for analysis, MAWI and DARPA 99. These data sets are composed of FTP, SSH, Telnet, Mail, DNS and HTTP traffic. For their experimental analysis, a comparison of this modified version of the auto-encoder neural network with the traditional neural network is conducted and the results show that although there is no significant difference when using a balanced dataset, the modified auto-encoder outperformed the traditional neural network when presented with

imbalanced data.

Semi-supervised Network Traffic Classification using Deep Generative Models

In [40], Li et. al. present a deep generative model to encode data into a lower dimensional feature space for network traffic classification. A variational auto-encoder (VAE) is created using a multilayer perceptron (MLP) which takes the flow features and transforms them into representation features with the purpose of finding the underlying structure. This is a completely unsupervised procedure. These representation features are acted on by a second generative model. This model is semi-supervised as it handles labeled and unlabeled data. Labeled data is used to tune the model by calculating the loss function in this second encoding model. Unlabeled data obtains its classification prediction as part of this second encoding. A stochastic gradient variational Bayes (SGVB) method is applied to optimize both VAEs. This model is tested against 4 distinct data sets:

- USTC Malware Traffic - Contains Cridex, Geodo, Htbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut and Zeus malware flows.
- USTC Normal Traffic - Contains BitTorrent, FaceTime, FTP, Gmail, MySQL, Outlook, Skype, SMB, Weibo and WorldOfWarcraft network flows.
- ISCX VPN - Contains Chat, Email, File, P2P, Streaming and VoIP data.
- USTC Anomaly Detection - A series of flows labeled as normal or malware.

For the experimental analysis of this generative model, the training and testing data is split with a 10:1 ratio. The training data is further subdivided into labeled and unlabeled data sets. The authors vary the number of labeled flows per class by selecting 20, 50, 100 and 200 flows per class. Experiments are conducted for each value of labeled flows with each of the 4 data sets. The results conclude that 20 labeled flows produced over 85% precision across all data sets, 50 labeled flow increased that to over 90% precision and 200 labeled flows reached over 95% precision. An interesting conclusion drawn from these experiments

is that the difference between normal and malware flows is easily detectable using this approach, as all values of labeled flows generated a perfect score of 100% for precision, recall and F1-score. Finally, the authors note that, when comparing to similar work which also uses a limited number of labeled flows for classification, others report an average of 80% precision, 75% recall and 76% F1-score. Those results are considerably lower than the results reported in this work.

Seq2Img: A Sequence-to-Image based Approach Towards IP Traffic Classification using Convolutional Neural Networks

In [41], Chen et. al. present a neural network approach to classifying network traffic. A Reproducing Kernel Hilbert Space (RKHS) is used in this approach as it allows for a compact way in which to represent conditional distributions. RKHS is applied to the 28 input features representing the first 10 packets of a flow:

- Packet size difference sequence (9 elements)
- Packet inter-arrival time sequence (9 elements)
- Packet direction sequence (9 elements)
- Server IP address

Once these features are transformed using RKHS, the resulting data is analyzed by a Convolutional Neural Network with two layers each with softmax pooling, and 3 fully connected layers that provide the final output. This neural network is compared with SVM, MLP, Naive Bayes and Decision Tree classifiers across two separate data sets. In the first data set containing FTP, HTTP, SSH, FTP and TLSV protocols, all classifiers as well as the neural network provide similar results and all are able to classify with above 90% accuracy. On the second data set containing Instagram, Skype, Facebook, Wechat and YouTube application data, the proposed CNN outperformed by a wide margin presenting 88.42% accuracy. The next closest being SVM at 76.93% accuracy and the remaining

classifiers all falling below 55%. This makes for a very interesting finding which should be further explored with larger data set and a wider variety of application data.

A novel QUIC traffic Classifier based on Convolutional Neural Networks

Tong et. al. provide interesting findings in [42]. A CNN classifier is combined with a Random Forest and both flow and packet data are used in this intriguing approach. As QUIC protocol traffic is analyzed, the authors note that there are several applications using this protocol to transmit information on the network. Voice call, chat, video streaming, Google play music and file transfer are all present in QUIC traffic. The authors note that upon inspection of packet sizes for each of the applications using QUIC, a pattern is found when considering the percentage of small packets (0 to 150 bytes) vs. large packets ($>$ 1000 bytes). The flow features used in this approach are:

- average payload length in both directions
- percentage of small bytes (0 to 150) in both directions
- percentage of medium bytes (150 to 1000) in both directions
- percentage of large bytes ($>$ 1000) in both directions

These flow features are used on the Random Forest classifier in order to find chat and Google Hangout voice call. This is the first stage of the classifier. In the second stage, packet data, consisting of the encrypted payload of the QUIC packets, is analyzed in addition to the flow features. Because the feature space must be 1400 bytes for the CNN, any packet payloads smaller than that are padded with zeros before analyzed. The byte sequence representing the packet payload is normalized by having all values divided by 255 resulting in numerical sequences between 0 and 1. The CNN consists of a convolutional layer, an average pooling layer and finally a fully connected layer. This converts the 1400 features into 3 classes representing the file transfer, video streaming and Google play music applications. The results of using this combination of classifiers produces a reported precision and recall score

of approximately 99%. Although the authors do note that this is not adequate for use in a real-time environment as presently constructed, this is a very interesting combination of features and classification techniques which show very promising results.

2.4.5 Comparison Table

Table 2.1: Comparison of related works.

Source	Algorithm/Model for classification	Features	Applications/Categories
BLINC [7]	graphlets	source IP/port destination IP/port average packet size (per flow)	Web P2P Data Network Management Mail News Chat Streaming Gaming Nonpayload Unknown
Traffic Classification Based on Zero-Length Packets [20]	Feature Extraction	Accumulated Application Protocol Data Units (a-APDU)	Amazon, Unknown, Yahoo, POP3, IRC, IMAPS, Telnet, HTTP-Proxy, MSN, Facebook, NFS, FTP-DATA, Oscar, Flash, WindowsUpdate, AppleiTunes, YouTube, SOCKS5, IAX, CiscoVPN, Dropbox, RDP, eDonkey, SSL, FTP-CONTROL, IPsec, SSH, Gmail, SMB, Google, HTTP, Skype, Oracle, OpenVPN, Wikipedia, GoogleMaps, POPS, Whois-DAS, PostgreSQL, MySQL, BitTorrent, H323, DNS, LastFM, CiscoSkinny, VNC, UPnP, eBay, Apple, Twitter, CNN, RTMP, Kerberos, IMAP
Machine Learning in Software Defined Networks: Data Collection and Traffic Classification [21]	Random Forest Stochastic Gradient Descent Extreme Gradient Descent	Packet size (first 5 packets) Packet time stamp (first 5 packets) Inter-arrival time (first 5 packets) source/destination MAC address source/destination IP address source/destination port number flow duration byte count packet count	BitTorrent Dropbox Facebook HTTP LinkedIn Skype Vimeo YouTube
SVM-based Classification Mechanism and Its Application in SDN Networks [22]	Support Vector Machine Decision Tree	Radial Basis Function kernel used to select feature set feature list not provided	Facebook, Line, YouTube, Skype, Google page, BitTorrent, Twitch, League of Legends, Messenger, Google Hangout, Spotify, Instagram, Dropbox, KKBOX, Sanguosha, MoPTT, PPS, WooTalk, IRC, PPLIVE, OneDrive, Yahoo page, Garena Messenger, Foxy, eDonkey, QQ, Pokemon Go
Network Traffic Classification Techniques and Comparative Analysis using Machine Learning Algorithms [23]	C4.5 (Decision Tree) Support Vector Machine BayesNet NaiveBayes	23 features (no feature list present)	WWW DNS FTP P2P Telnet
A Comparative Performance Analysis on Network Traffic Classification using Supervised Learning Algorithms [24]	Naive Bayes BayesNet Complement Naive Bayes Decorate Random Forest Bagging AdaBoost Stacking Rotation Forest	266 features total, subset selected: First quartile inter-arrival time Freq. of zero receive window SYN permission of SACK Total time for data transfer Number of RTT samples found Median Ethernet frame bytes Min. arrival time between packets	WWW Mail FTP Multimedia Games P2P
Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification [25]	K Nearest Neighbor Linear Discriminant Analysis (LDA)	Average packet size flow duration bytes (per flow) packets (per flow) Root Mean Squared of packet size	Interactive (Telnet) bulk data (FTP data, Kazaa) streaming (RealMedia streaming) transactional (DNS, HTTPS)

Source	Algorithm/Model for classification	Features	Applications/Categories
Toward Classifying Unknown Application Traffic [26]	K Nearest Neighbor with Kolmogorov-Smirnov distance	Packet count Byte count Average packet inter-arrival time Average bit rate of connection Largest/Smallest packet size Min/Max packet inter-arrival time Direction (inbound/outbound) Number of ARP packets Number of DNS packets Number of TCP ACKs Min/Max receive window	YouTube Google Hangout Skype HTTP Web browsing
Early Classification of Residential Networks Traffic using C5.0 Machine Learning Algorithm [27]	C5.0 C4.5 K Nearest Neighbor Naive Bayes	Payload size of first 10 packets Direction of first 10 packets Packet inter-arrival time Inter Downlink packet arrival time processed packets count	BitTorrent Facebook Google-Services Web browsing Secure Web browsing QUIC Skype
Network Traffic Classification based on Transfer Learning [28]	TrAdaBoost AdaBoost with Maxent	Server port Min packet inter-arrival time Mean packet inter-arrival time Variance packet inter-arrival time Mean packet bytes per flow Mean control bytes per flow Average window size Mean IP packet bytes	WWW Mail Database FTP-data P2P Services
PSO Optimized Semi-Supervised Network Traffic Classification Strategy [29]	C4 NBK KMKNN PSO-KNN	Utilized RSEC to select features (feature list not provided)	WWW FTP RTX Bulk data RTMP
A Framework for QoS-aware Traffic Classification Using Semi-supervised Machine Learning in SDNs [30]	Laplacian SVM	src to dest entropy of packet length dest to src entropy of packet length source/destination port src to dest avg packet length dest to src avg packet length packets to respond src to dest min packet length dest to src packet interactivity src to dest median packet length src to dest	Voice/Video Conference (Skype,QQ, Google Hangout, ...) Interactive Data (Gaming, Web, HTTP services, ...) Streaming (PPStream, Vimeo, SopCast, ...) Bulk Data Transfer (FTP, Torrent, Dropbox, ...)
QoS-aware Traffic Classification Architecture Using Machine Learning and Deep Packet Inspection in SDNs [31]	Support Vector Machine Bayes Classifier K Nearest Neighbor	Data flow time characteristics Packet characteristics Protocol characteristics Hurst parameter	Voice (Skype, QQ, WeChat...) Video (YouTube, Youku, Vimeo...) Bulk Data (FTP, Dropbox, Torrent...) Interactive (LOL, Dota, HTTP...)
A Framework & System for Classification of Encrypted Network Traffic using Machine Learning [32]	Bagging of: Logistic Regression Support Vector Machine Decision Tree Adaboost Neural Networks Naive Bayes	Flow duration Protocol Packet, byte and payload count Entropy of packet size min, max, mean and var of: -packet size -packet inter-arrival time	Video Streaming (YouTube, Netflix) Video Chat (Skype, Messenger) Audio Stream (Spotify, SoundCloud) VoIP (Skype, Messenger) File Transfer (Dropbox, Google Drive) Mail (Gmail, Yahoo) Web browsing (Firefox, Chrome) P2P (BitTorrent, eDonkey) Chat Message (Facebook, Telegram) ToR Traffic (Video Streaming, Web)
A New Semi-supervised Method for Network Traffic Classification Based on X-means Clustering and Label Propagation [33]	X-means clustering K-nearest neighbor J48 Naive Bayes	42 features in Moore data set feature list not provided	Bulk (ftp) Database (posture, silent oracle, ingres) Interactive (ssh, klogin, rlogin, telnet) Mail (imap, pop2/3, smtp) Services (X11, dns, ident, Idap, ntp) WWW P2P (KaZaA, BitTorrent, GnuTella) Attack (worm and virus attacks) Games (Half-Life) Multimedia (Win Media Player, real)
Packet-based Network Traffic Classification Using Deep Learning [34]	CNN ResNet	Image representation of bits in flows by converting payload data into 4 bit pixels and generating 6x6, 8x8, 16x16 and 32x32 images	RDP Skype SSH BitTorrent HTTP-Facebook HTTP-Google HTTP-Wikipedia HTTP-Yahoo

Source	Algorithm/Model for classification	Features	Applications/Categories
A Traffic Classification Method Based on Packet Transport Layer Payload by Ensemble Learning [35]	Bagging of: 1DCNN 2DCNN LSTM	First 784 bytes of the payload	Email (gmail,POP3,SMPT,IMAP) Chat (ICQ, AIM, Skype, Facebook, Hangouts) Stream (Vimeo, Youtube, Netflix, Spotify) P2P (uTorrent, BitTorrent) VoIP (Facebook, Skype, Hangouts, Voipbuster) File Transfer (Skype, FTPS, SFTP)
Byte Segment Neural Network for Network Traffic Classification [36]	Recurrent neural networks LSTM GRU Focal loss SVM C4.5 BayesNet	Packet payload binaries	DNS BitTorrent PPLive QQ SMTP 360 Amazon Yahoo Cloudmusic Foxmail
Leveraging Inner-Connection of Message Sequence for Traffic Classification: A Deep Learning Approach [37]	LSTM neural network	message sequence extracted from traffic behavior	HTTP, SSH, SMTP, WeChat, Remote Desktop, P2P, Video
Common Knowledge Based Transfer Learning of Traffic Classification [38]	Multi-output Deep Neural Network (DNN)	16 features selected from WITS ISPDSL-I, ISPDSL II, Entry09 and Entry10 feature list not provided	Application list not provided
An Improved Stacked Auto-Encoder for Network Traffic Flow Classification [39]	Auto-encoder (Neural Network) with Bayesian probability training method	flows, specific metrics presented to algorithm omitted	FTP SSH Telnet Mail DNS HTTP
Semi-supervised Network Traffic Classification using Deep Generative Models [40]	MLP with Stochastic Gradient Variational Bayes (SGVB) for loss function optimization	Variational Auto-Encoder (VAE) generated mathematical model for feature representation in lower dimensional space	4 datasets: USTC Malware Traffic (Cridex, Geodo, Htbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut, Zeus) USTC Normal Traffic (BitTorrent, FaceTime, FTP, Gmail, MySQL, Outlook, Skype, SMB, Weibo, WorldOfWarcraft) ISCX VPN (Chat, Email, File, P2P, Streaming, VoIP) USTC Anomaly Detection (Normal, Malware)
Seq2Img: A Sequence-to-Image based Approach Towards IP Traffic Classification using Convolutional Neural Networks [41]	CNN MLP SVM Decision Tree Naive Bayes	First 10 packet of each flow: Packet size difference sequence Packet inter-arrival time sequence Packet direction sequence Server IP address	FTP, HTTP, SSH, FTP, TLSV Instagram, Skype, Facebook, WeChat, YouTube
A novel QUIC traffic Classifier based on Convolutional Neural Networks [42]	CNN Random Forest	features all in two directions: average payload length percentage of small, medium and large packets in the flow	All using QUIC protocol: Voice call, chat, video streaming, Google play music and file transfer

Chapter 3

Experimental Plan

In this chapter, two separate sets of experiments are conducted in order to evaluate the capacity of a set of flow features to provide network application classification. The objective of the first set of experiments is to understand the effectiveness of non-address inherent and derived flow features in classifying network traffic. As part of this experimental set, a rule-based expert system is derived from the port number conventions in order to generate the training data necessary for experimentation. The experimental plan for this initial effort is detailed in section 3.1. The objective of the second set of experiments is to uncover the classification performance of different combinations of flow features. In addition to using the inherent and derived features from the first experiment set, features describing host behavior are engineered. The classification power of all possible combinations of these engineered features is evaluated. Additionally, the training data for the second experimental set is produced using deep packet inspection during the data capture. This is done to obtain a closer approximation to the "ground truth" relative to the rule-based expert system used in the first experiment set. Section 3.2 describes the second experimental plan corresponding to the second experimental set.

3.1 Experiment Set 1: Can flow features achieve network application classification?

In the first experimentation set, the focus is on the ability of three non-address inherent and derived flow features to classify network traffic,

- Number of bytes - Inherent flow feature
- Number of packets - Inherent flow feature
- Flow duration - Derived flow feature

3.1.1 Rule-Based Expert System

The necessity for labeled flows to serve as training data for machine learning algorithms encouraged the creation of a rule-based expert system. This expert system generated labels with the assistance of port number conventions in a two step process.

1. Tuples from the port and IP protocol at both the source and destination of each network flow are created. These source and destination tuples are mapped to the registered port numbers in IANA which result in two labels, one for the source and another for the destination.
2. A comparison of the source and destination labels is performed:
    - If both sides are unknown (there is no mapping found in IANA for either port/IP protocol tuple), the resulting application label is unknown.
    - If one side is unknown, the resulting application label corresponds to the known side.
    - If both sides are known, the resulting application is the most likely application.

These generated labels will serve as our "ground truth" for the experimental analysis of the selected feature's ability to classify network traffic. Aware of the limitations involved with using port number conventions, additional measures are taken into consideration in selecting the training data from the set of flows labeled with the rule-based expert system. The applications selected are those who's port numbers are unlikely to be misused by another application. Additionally, only applications with many training samples are considered.

3.1.2 Experimental Setup

Once the method for generating training data is established, a set of experiments is conducted to better understand how classification performance is affected using the flow features selected while varying the:

- application subset: { Select 5, Top 4, Top 5, Top 10 },
- type of flows: { All flows, only large elephant flows },
- machine learning technique: { K nearest neighbors, Decision tree, Random forest },
- and machine learning hyper-parameters.

Data collected from the University of Kentucky campus network is used for the experimental analysis. The training data set is a composite of several 1 or 2 hour data sets collected over several days. The test data set is an 8-hour data set collected continuously on a single day. Both the training and testing data sets are appended with the application label according to the rule-based expert system. The resulting flow data is displayed in Table 3.1 which lists the Select 5 applications in the UKY data sets and their corresponding number of flows, while Table 3.2 lists the Top 10 applications in the training data set ordered by number of flows. Note the exclusion of HTTP and HTTPS since there is low confidence in the use of port number conventions for those applications.

The machine learning algorithms are implemented in Python 3 using several data analytics libraries (e.g., NumPy, PANDAS [43], sci-kit learn [44]). First, the network flow data is loaded into a PANDAS dataframe. Then begins the data preprocessing by first removing any flows where the application label is 'unknown' or 'na'. Then, any flows where there is a discreptancy in labels generated from source and destination, meaning that they are not labeled as the same application on both sides, are discarded as there is not enough confidence in these labels to represent "ground truth".

With the data preprocessed, the experimental analysis can take place. The first step is to filter the preprocessed data according to the specifics of the particular experiment by

Table 3.1: UKY Select 5 Applications with respective flow count.

	FLOWS	
APPS	**ALL**	**TOP 50%**
ALL	28136	14068
smtp	17233	7883
ssh	8894	4678
domain	1155	659
snmp	849	848
telnet	5	N/A

Table 3.2: UKY Top 10 Applications ordered by flow count.

	FLOWS	
APPS	**ALL**	**TOP 50%**
ALL	149121	74561
netbios-dgm	51934	34164
netbios-ns	33855	18731
smtp	17233	3983
bootps	10820	9217
microsoft-ds	8930	772
ssh	8894	1009
ntp	6840	N/A
mdns	4733	2663
hsrp	3391	2848
syslog	2491	1174

selecting the adequate subset of applications (Select 5, Top 4, Top 5, Top 10) and subset of flows (All, Top 50% elephant flows). Having selected the flow data specific to the experiment being conducted, the feature space (# of bytes, # of packets and duration) and target (generated application labels) are separated from the rest of the flow data. The machine learning model is created and the particular model's hyper-parameters are selected with the assistance of sci-kit learn's GridSearchCV() which runs a cross-validated grid-search over a specified parameter grid and returns the optimal hyper-parameters. The model is then fitted with the training set and subsequently evaluated with the test set. All pertinent information on the experiment, including model's hyper-parameters and evaluation performance metrics, are reported and saved onto a csv file. This is conducted for all machine learning algorithms on all the possible data sets generated from the variations previously described [45].

3.2 Experiment Set 2: What is the classification performance of combinations of flow features?

Recalling from Section 2.1.3, there are inherent, derived and engineered flow features. Having previously conducted experimental analysis on a subset of inherent and derived features, the next logical step is to generate interesting new features and evaluate their ability to classify network traffic. The work described in this section follows this approach.

3.2.1 Flow Features

The work presented by Karagiannis et. al. in [7] provides an interesting approach to network traffic classification. Their focus is in associating network hosts with the applications being used based on the communication behavior. Once the behavior is represented in graphlets, they use these graphlets to classify unlabeled graphlets representing new traffic. Three numerical features are engineered based on BLINC's behavioral analysis of network

traffic.

Destination Address Count

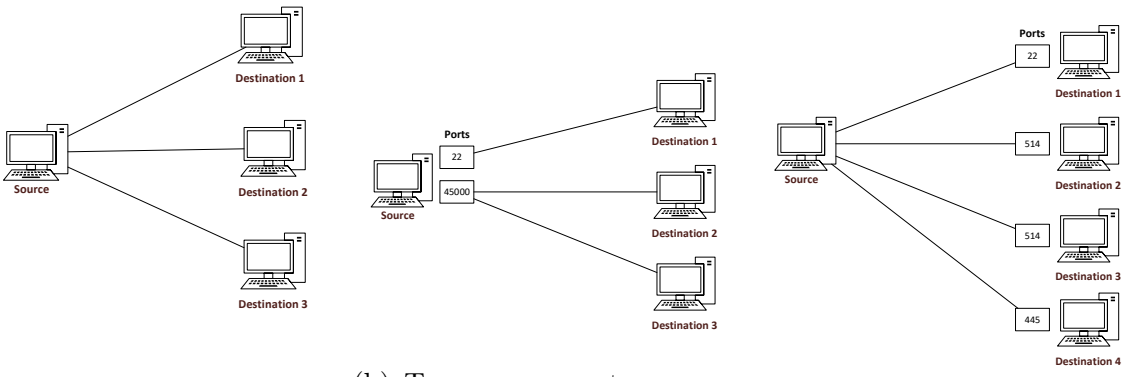
The number of unique destination addresses each source address will communicate with throughout the data set is analyzed. As explained in [7], this is representative of the source address "popularity". This communication with few or many different destination addresses creates a quantitative metric which provides host level behavioral context. The idea is that this translates an address level interaction to a value which can then be leveraged with the machine learning techniques to better classify network applications. Figure 3.1a would be numerically represented as $dstaddrcount = 3$ as this particular source address is communicating with three different destination hosts on their respective addresses.

Source and Destination Port Count

The port numbers provide insight into the functional behavior of the hosts represented in the flow data. The number of unique ports the source host is using to communicate is evaluated first. Not considering which source port numbers are used, the importance here is with the total number of unique source ports being used by each source address. As seen in Figure 3.1b, $srcportcount = 2$ as this particular source address is using two source ports.

Similarly, the number of unique destination ports that a source address is communicating with across all its destination addresses is investigated. In terms of Figure 3.1c, there are three unique destination port numbers across 4 destination addresses for the source host. This would give us the feature $dstportunique = 3$.

In this work, the analysis of the original 3 features (# of bytes, # of packets and duration) from the previous work is improved by the inclusion of these new engineered features (dstaddrcount, srcportcount and dstportunique) and their classification power is evaluated.



(a) Source host communicating with three unique destination hosts

(b) Two source ports communicating with destination hosts

(c) Source communicating with destination ports

Figure 3.1: BLINC-inspired engineered features.

3.2.2 Experimental Setup

The network flow data for these experiments is collected once again from the University of Kentucky (UKY). The collection instrument is attached to a port which monitors a border aggregation switch using a 100G Ethernet connection. This aggregation switch is the link between border routers, distribution routers, remote data centers (private links), an SDN-enabled science Demilitarized Zone (DMZ), High Performance Computing Cluster (HPC), Data Transfer Nodes (DTNs), and cloud DTNs used by clinical and research laboratories. The upstream ports of the aggregation switch connecting various devices and networks are replicated to the 100G monitoring port, providing aggregate visibility of data being transmitted between networks, but not within those networks that are downstream. This over-subscribed aggregation of high-speed links creates the possibility that packets will be dropped if the total capacity of links exceeds 100G. However, outside of experimental testing 100G in aggregate traffic has not been observed. In addition, operational flow rates that exceeded the capacity of the measurement instrument to accurately generate network flow data from raw monitored traffic have also not been observed. Figure 3.2 provides an illustration of the measurement instrument deployment.

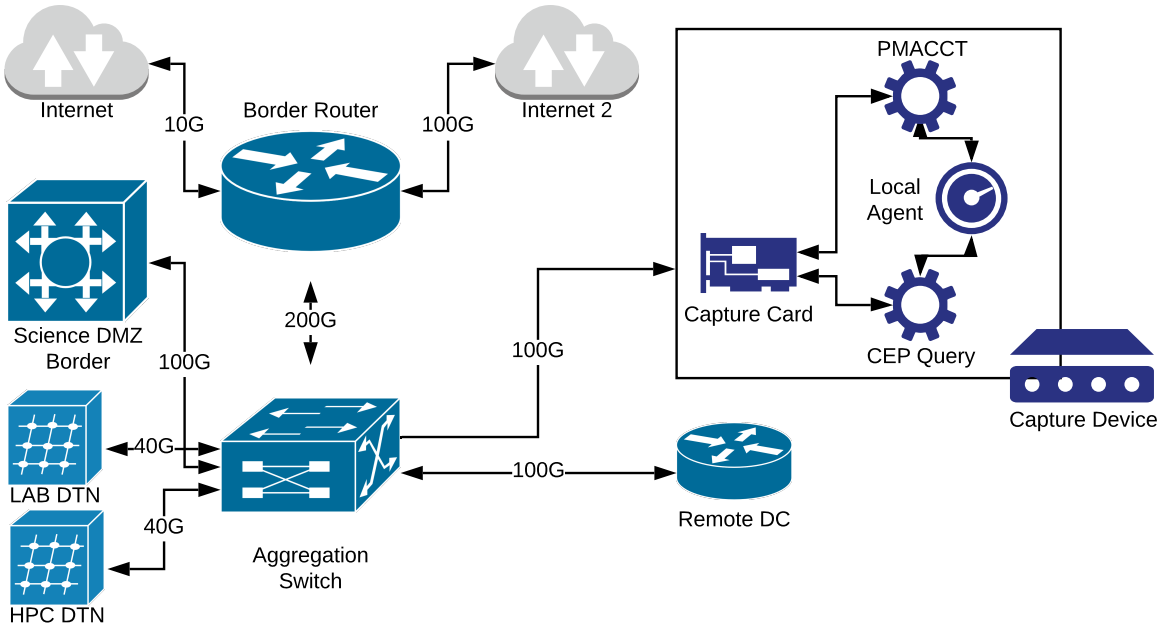


Figure 3.2: Network measurement instrument used to collect flow data.

Data is captured via the collection instrument by leveraging pmacct[17], an open-source passive network monitoring tool. Pmacct is enhanced with nDPI[15], an open-source deep packet inspection library, to generate network application labels which will serve as "ground truth" for our experimental analysis. A series of 1, 4 or 24 hour flow captures is conducted spanning 17 days and a combined 361 hours of network traffic data, including weekdays and weekends at different times of day (morning, afternoon and evening). This is done to account for changes in network behavior due to temporal bias. Each individual 1, 4 or 24 hour flow capture is then appended with the derived flow duration and augmented by the BLINC inspired features (*dstaddrcount*, *srcportcount*, and *dstportunique*). A composite data set is created by aggregating all the collected UKY flow data. This composite data set is reduced by discarding any flows with a zero duration as any flows with such a short life are deemed insignificant and thus not considered. At this point, similar nDPI-created network application labels are aggregated into classes according to the type of service provided. Table 3.3 displays the breakdown of each class created and their constituent nDPI application labels.

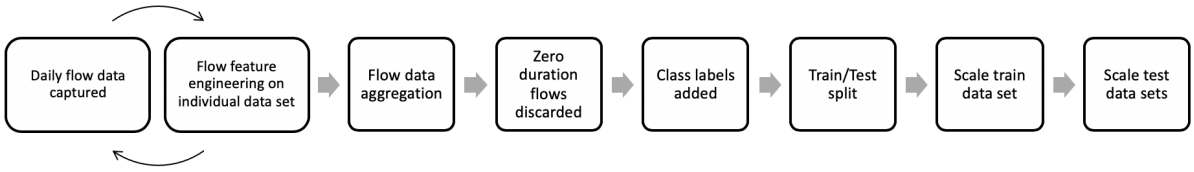


Figure 3.3: Data preparation pipeline

Once the data preprocessing is complete, it must be split into subsets for training and testing. This is performed using stratified sampling, to account for the imbalanced nature of the classes in the data set, and a 2:1 train/test split. With the test subset being further split into 6 test subsets. Once this is completed for all classes, the resulting subsets from each class are combined to form 6 test sets with similar distributions. Table 3.4 provides the breakdown of the flow data into the train and 6 test subsets.

Table 3.3: Class labels and associated nDPI application labels.

Class	nDPI application label
Authentication	Kerberos
Big Tech	Amazon
	Apple
	Google
	Microsoft
	Playstore
	UbuntuONE
Chat_VoIP	GoogleHangout
	IRC
	Oscar
	QQ
	QUIC
	RTMP
	SIP
	STUN
	Skype
	SkypeCall
	TeamSpeak
	Viber
File Transfer	BitTorrent
	FTP_CONTROL
	FTP_DATA
	NFS
Github	Github
HTTP	HTTP
HTTPS	SSL
	SSL_No_Cert
M2M_Messaging	ApplePush
	GoogleServices
	MQTT
Network	SNMP
Management	Syslog
	Whois-DAS

Class	nDPI application label
Network	BGP
Operation	Cloudflare
	DHCPV6
	DNS
	ICMP
	ICMPV6
	IGMP
	mDNS
	NTP
	SSDP
	Teredo
	UPnP
Remote login	RDP
	SSH
Video Streaming	YouTube
Unknown	AppleiCloud
	COAP
	Facebook
	Gmail
	GoogleDocs
	GoogleDrive
	GoogleMaps
	LinkedIn
	MS_OneDrive
	NetBIOS
	Office365
	Redis
	Tor
	Twitter
	Unknown

Table 3.4: Train and Test sets breakdown by class label

	Train	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
Authentication	117	10	10	10	10	10	9
Big Tech	173422	14237	14237	14236	14236	14236	14236
Chat VoIP	18147	1490	1490	1490	1490	1490	1489
File Transfer	235	20	20	20	19	19	19
Github	481	40	40	40	40	39	39
HTTP	7363	605	605	605	605	604	604
HTTPS	73841	6062	6062	6062	6062	6061	6061
M2M Messaging	12492	1026	1026	1026	1025	1025	1025
Network Management	370414	30408	30408	30407	30407	30407	30407
Network Operation	1501297	123241	123241	123241	123241	123241	123240
Remote Login	1337029	109757	109756	109756	109756	109756	109756
Video Streaming	902	75	74	74	74	74	74
Unknown	2333153	191528	191528	191528	191528	191527	191527

Having split the data into training and testing sets, data standardization is conducted. This consists of transforming each feature distribution to one with zero mean and unit variance. This is common practice when applying many machine learning models as this eliminates the potential bias caused by the difference in magnitudes among the distinct features. First, the training data set is analyzed to find the adequate mean and standard deviation values which will be used to transform this data. Once computed, the training data set is transformed. All 6 test sets are then individually transformed using the training data transformation values. Again, this is common practice as the machine learning model should make a prediction based solely on the training data [46]. This finalizes our data preparation pipeline, visualized in Figure 3.3.

The experimental analysis in this work consists of implementing the same machine learning algorithms as in the previous experimentation set (k nearest neighbor, decision tree and random forest) for consistency. For the feature selection, the original features from the initial experiments (# of bytes, # of packets and duration) are always included in the feature set. All the possible combinations of the three engineered features (dstaddrcount, srcportcount and dstportunique) are added to the original features, resulting in 8 possible

combinations of flow features. Training and testing data is filtered according to the feature combination selected and then separated into the feature and target space. The machine learning models are created, hyper-parameters tuned using sci-kit learn's GridSearchCV() as in the previous work, fitted with the training data and then each trained model is evaluated with all 6 test subsets with the resulting evaluation metrics being saved for analysis.

Chapter 4

Experimental Results

4.1 Experiment Set 1: Can flow features achieve network application classification?

After executing the first experimental plan, Table 4.1 shows the experimental results. Each row is the result of an experiment with the rows first grouped by the set of applications (Select 5, Top 4, Top 5, Top 10), followed by the flows included (all vs. Top 50% elephant), and finally by the machine learning technique (KNN, Decision Tree, Random Forest). For each row the results shown are: 1) dataset size after focusing on the particular set of applications, 2) best performing hyper-parameter values, 3) accuracy, 4) precision, and 5) recall.

The primary observations are:

1. A random forest classifier mostly provides the highest accuracy.
2. A decision tree classifier has performance very close to the random forest classifier.
3. Unexpectedly, focusing on elephant flows decreases accuracy slightly.
4. Overall, precision and recall measures are similar to accuracy.

Elephant flows are large byte and subsequently large packet and large duration flows. With the increased range of these three flow features (# of bytes, # of packets, and duration) the expectation was an increased distinction among different applications thereby leading to higher accuracy. However, focusing on elephant flows provides fewer flows to train

our machine learning algorithms and thus decreasing, rather than increasing, its accuracy slightly.

To gain some deeper insight into the classification performance we can examine the per-class measures. Precision, recall, and support are shown per-class for Select 5 and Top 10 in Tables 4.2 and 4.3 respectively. For Select 5, smtp and ssh reduce the overall accuracy, as noted by their recall values, despite the recall values for domain and snmp being rather high. It is not possible to classify telnet given the small number of samples. With the rarity of telnet flows, removing this application from consideration in future endeavors is appropriate.

For Top 10, recall is rather high for most applications, with ssh and smtp being exceptions. The results indicate that some applications can be classified using only the three non-address flow features found in NetFlow records while others likely require more information.

Table 4.1: Experimental classification results: We varied the application set (Select 5, Top 4, Top 5, Top 10), flow type (all, Top 50% elephant), and machine learning technique (KNN, Decision Tree, Random Forest).

	Flows	Machine Learning Technique	dataset size (flows)	best parameters	accuracy	precision	recall
SELECT 5	**ALL**	KNN	28136	n neighbors: 4	0.61043	0.76	0.61
		DecisionTree	28136	criterion : gini max depth: 16 max features: 3	0.65966	0.83	0.66
		RandomForest	28136	criterion : entropy max depth: 16 max features: 3 n estimators: 64	0.65864	0.83	0.66
	TOP 50%	KNN	14068	n neighbors: 64	0.74651	0.75	0.75
		DecisionTree	14068	criterion : entropy max depth: 8 max features: 3	0.83791	0.89	0.84
		RandomForest	14068	criterion : entropy max depth: 8 max features: 3 n estimators: 64	0.84058	0.88	0.84
TOP 4	**ALL**	KNN	113842	n neighbors: 2	0.90961	0.92	0.91
		DecisionTree	113842	criterion : entropy max depth: None max features: 3	0.99139	0.99	0.99
		RandomForest	113842	criterion : entropy max depth: None max features: 3 n estimators: 64	0.99251	0.99	0.99
	TOP 50%	KNN	56921	n neighbors: 64	0.68587	0.68	0.69
		DecisionTree	56921	criterion : entropy max depth: None max features: 3	0.96071	0.97	0.96
		RandomForest	56921	criterion : entropy max depth: None max features: 3 n estimators: 32	0.96277	0.97	0.96
TOP 5	**ALL**	KNN	122772	n neighbors: 4	0.87294	0.88	0.87
		DecisionTree	122772	criterion : entropy max depth: None max features: 3	0.95840	0.96	0.96
		RandomForest	122772	criterion : entropy max depth: None max features: 3 n estimators: 64	0.96070	0.96	0.96
	TOP 50%	KNN	61386	n neighbors: 64	0.68838	0.68	0.69
		DecisionTree	61386	criterion : entropy max depth: None max features: 3	0.93873	0.95	0.94
		RandomForest	61386	criterion : gini max depth: 16 max features: 3 n estimators: 8	0.93165	0.95	0.93
TOP 10	**ALL**	KNN	149121	n neighbors: 2	0.81978	0.83	0.82
		DecisionTree	149121	criterion : entropy max depth: None max features: 3	0.91080	0.93	0.91
		RandomForest	149121	criterion : entropy max depth: 16 max features: 3 n estimators: 64	0.91211	0.94	0.91
	TOP 50%	KNN	74561	n neighbors: 64	0.67276	0.67	0.67
		DecisionTree	74561	criterion : entropy max depth: 8 max features: 3	0.87972	0.87	0.88
		RandomForest	74561	criterion : entropy max depth: 8 max features: 3 n estimators: 32	0.88160	0.87	0.88

Table 4.2: Per-class classification results (Select 5, Random Forest).

	SELECT 5					
	ALL			**TOP 50%**		
Application	**precision**	**recall**	**support**	**precision**	**recall**	**support**
domain	0.79	0.95	1065	0.99	0.98	679
smtp	0.92	0.63	20343	0.95	0.84	9487
snmp	0.92	0.95	914	0.91	0.82	913
ssh	0.22	0.65	3162	0.46	0.78	1667
telnet	0.00	0.00	8	N/A	N/A	N/A

Table 4.3: Per-class classification results (Top 10, Random Forest).

	TOP 10					
	ALL			**TOP 50%**		
Application	**precision**	**recall**	**support**	**precision**	**recall**	**support**
bootps	0.96	0.98	11141	0.80	0.74	9472
hsrp	0.97	0.98	3205	0.96	0.89	2705
mdns	0.91	0.95	2997	0.73	0.48	2130
microsoft-ds	0.79	0.75	11487	0.65	0.09	1110
netbios-dgm	0.99	0.99	55624	0.88	0.97	34876
netbios-ns	1.00	1.00	36384	0.95	0.99	19340
ntp	0.99	1.00	7250	N/A	N/A	N/A
smtp	0.85	0.61	20343	0.92	0.70	5185
ssh	0.21	0.60	3162	0.46	0.52	968
syslog	0.86	0.79	2660	0.41	0.17	1341

4.2 Experiment Set 2: What is the classification performance of combinations of flow features?

Once our second classification experiment set concludes, the precision and recall is reported for each combination and each machine learning technique used along with their corresponding 99% confidence intervals to demonstrate statistical significance. The results of this comparison can be seen in Tables 4.4, 4.5 and 4.6. These results are also visualized in Figure 4.1 where the precision and recall confidence intervals for each feature set are displayed by machine learning algorithm. Additionally a breakdown showing the precision scores for each of the feature combinations with each machine learning technique at the class level are shown in Figures 4.2 to 4.9.

Table 4.4: KNN experimental results.

	Test 1		Test 2		Test 3		Test 4		Test 5		Test 6		Confidence Interval 99%	
Features	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
duration, dPkts, dOctets	0.9536	0.9541	0.9539	0.9545	0.9534	0.9539	0.9530	0.9536	0.9532	0.9539	0.9536	0.9542	0.9535 ± 0.0008	0.9540 ± 0.0007
+ (dstaddrcount)	0.9365	0.9375	0.9367	0.9376	0.9363	0.9372	0.9364	0.9374	0.9365	0.9376	0.9361	0.9370	0.9364 ± 0.0005	0.9374 ± 0.0006
+ (srcportcount)	0.9213	0.9226	0.9220	0.9232	0.9207	0.9220	0.9210	0.9224	0.9215	0.9229	0.9208	0.9221	0.9212 ± 0.0012	0.9225 ± 0.0011
+ (dstportunique)	0.9323	0.9333	0.9329	0.9338	0.9322	0.9332	0.9319	0.9329	0.9327	0.9337	0.9321	0.9330	0.9324 ± 0.0009	0.9333 ± 0.0009
+ (dstaddrcount, srcportcount)	0.9321	0.9328	0.9321	0.9326	0.9313	0.9319	0.9316	0.9323	0.9315	0.9323	0.9315	0.9322	0.9317 ± 0.0008	0.9324 ± 0.0008
+ (dstaddrcount, dstportunique)	0.9374	0.9382	0.9375	0.9383	0.9370	0.9379	0.9370	0.9379	0.9374	0.9384	0.9368	0.9377	0.9372 ± 0.0007	0.9381 ± 0.0007
+ (srcportcount, dstportunique)	0.9271	0.9275	0.9275	0.9278	0.9267	0.9270	0.9266	0.9271	0.9270	0.9275	0.9266	0.9270	0.9269 ± 0.0009	0.9273 ± 0.0008
+ (dstaddrcount, srcportcount, dstportunique)	0.9322	0.9330	0.9323	0.9330	0.9318	0.9326	0.9318	0.9327	0.9319	0.9329	0.9317	0.9326	0.9320 ± 0.0006	0.9328 ± 0.0005

Table 4.5: DT experimental results.

	Test 1		Test 2		Test 3		Test 4		Test 5		Test 6		Confidence Interval 99%	
Features	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
duration, dPkts, dOctets	0.8782	0.8836	0.8780	0.8839	0.8785	0.8843	0.8776	0.8830	0.8774	0.8833	0.8774	0.8831	0.8779 ± 0.0011	0.8835 ± 0.0012
+ (dstaddrcount)	0.9368	0.9392	0.9367	0.9392	0.9364	0.9390	0.9367	0.9390	0.9373	0.9394	0.9364	0.9388	0.9367 ± 0.0008	0.9391 ± 0.0005
+ (srcportcount)	0.9310	0.9331	0.9323	0.9336	0.9316	0.9333	0.9306	0.9326	0.9318	0.9335	0.9311	0.9332	0.9314 ± 0.0015	0.9332 ± 0.0009
+ (dstportunique)	0.9296	0.9315	0.9295	0.9316	0.9291	0.9313	0.9290	0.9310	0.9302	0.9319	0.9292	0.9312	0.9294 ± 0.0011	0.9314 ± 0.0008
+ (dstaddrcount, srcportcount)	0.9353	0.9370	0.9351	0.9373	0.9349	0.9367	0.9352	0.9373	0.9343	0.9368	0.9353	0.9374	0.9350 ± 0.0009	0.9371 ± 0.0007
+ (dstaddrcount, dstportunique)	0.9499	0.9519	0.9505	0.9522	0.9502	0.9521	0.9498	0.9517	0.9508	0.9525	0.9502	0.9518	0.9502 ± 0.0009	0.9520 ± 0.0007
+ (srcportcount, dstportunique)	0.9535	0.9553	0.9542	0.9557	0.9536	0.9551	0.9534	0.9550	0.9545	0.9558	0.9542	0.9553	0.9539 ± 0.0011	0.9554 ± 0.0008
+ (dstaddrcount, srcportcount, dstportunique)	0.9612	0.9629	0.9617	0.9635	0.9611	0.9628	0.9605	0.9624	0.9617	0.9634	0.9615	0.9633	0.9613 ± 0.0011	0.9631 ± 0.0010

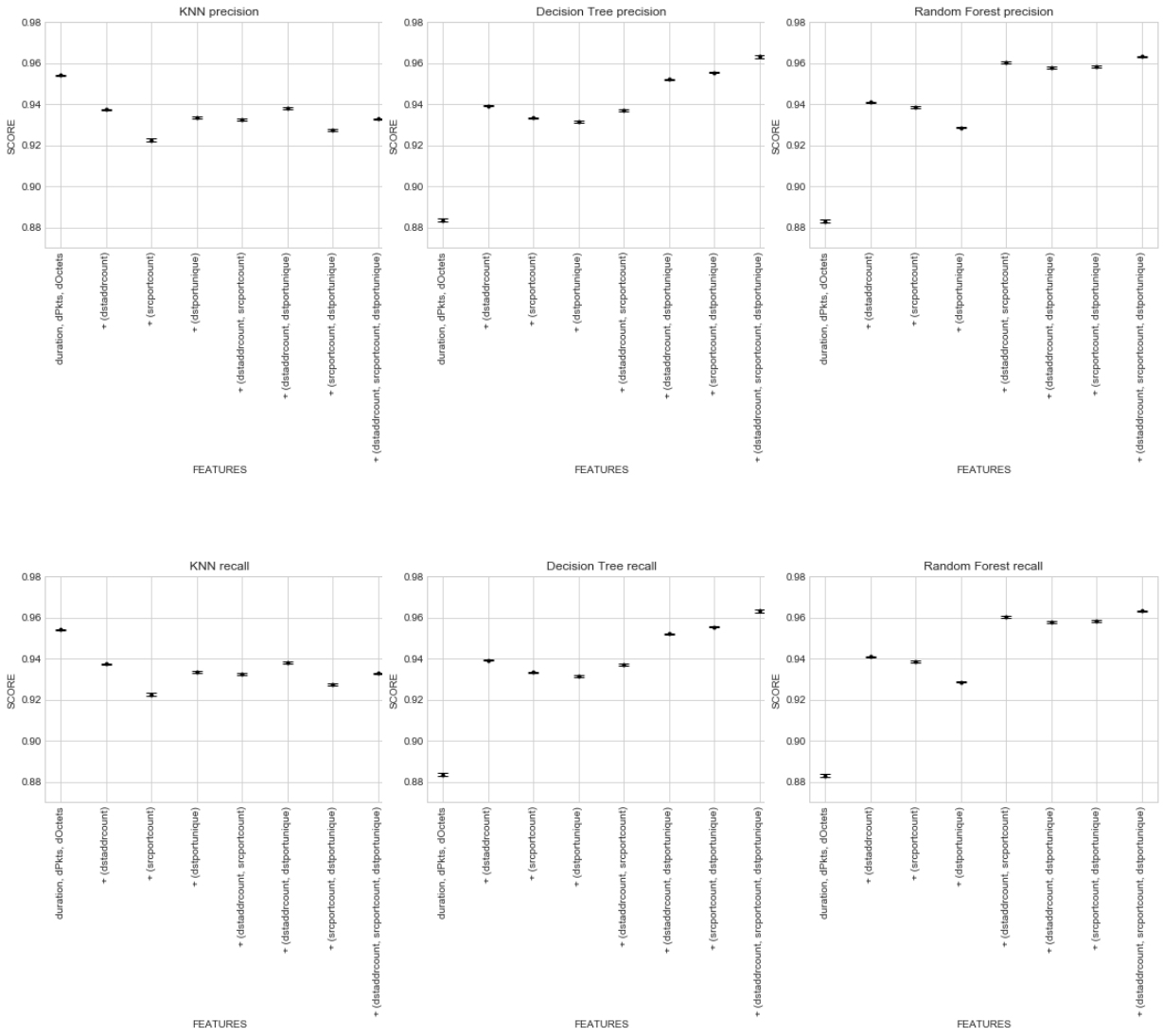


Figure 4.1: Experimental results by machine learning technique with confidence interval.

Table 4.6: RF experimental results.

	Test 1		Test 2		Test 3		Test 4		Test 5		Test 6		Confidence Interval 99%	
Features	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
duration, dPkts, dOctets	0.8769	0.8831	0.8771	0.8833	0.8777	0.8839	0.8786	0.8826	0.8779	0.8830	0.8764	0.8826	0.8774 ± 0.0019	0.8831 ± 0.0012
+ (dstaddrcount)	0.9391	0.9410	0.9395	0.9412	0.9396	0.9408	0.9401	0.9409	0.9401	0.9412	0.9399	0.9408	0.9397 ± 0.0009	0.9410 ± 0.0004
+ (srcportcount)	0.9366	0.9382	0.9379	0.9390	0.9373	0.9382	0.9349	0.9376	0.9359	0.9385	0.9370	0.9385	0.9366 ± 0.0026	0.9383 ± 0.0011
+ (dstportunique)	0.9270	0.9286	0.9268	0.9287	0.9265	0.9287	0.9269	0.9282	0.9284	0.9292	0.9258	0.9282	0.9269 ± 0.0021	0.9286 ± 0.0009
+ (dstaddrcount, srcportcount)	0.9587	0.9602	0.9597	0.9610	0.9586	0.9601	0.9588	0.9600	0.9597	0.9609	0.9593	0.9605	0.9591 ± 0.0012	0.9604 ± 0.0010
+ (dstaddrcount, dstportunique)	0.9558	0.9575	0.9564	0.9580	0.9557	0.9574	0.9555	0.9574	0.9563	0.9580	0.9560	0.9577	0.9559 ± 0.0008	0.9577 ± 0.0007
+ (srcportcount, dstportunique)	0.9574	0.9578	0.9581	0.9585	0.9574	0.9579	0.9575	0.9576	0.9578	0.9583	0.9581	0.9583	0.9577 ± 0.0008	0.9581 ± 0.0008
+ (dstaddrcount, srcportcount, dstportunique)	0.9621	0.9630	0.9625	0.9635	0.9621	0.9631	0.9620	0.9629	0.9623	0.9634	0.9622	0.9634	0.9622 ± 0.0004	0.9632 ± 0.0006

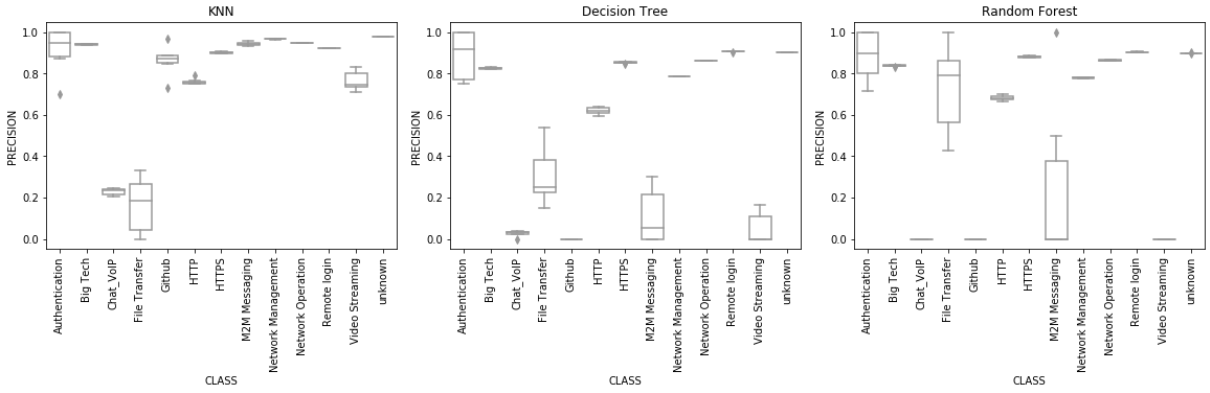


Figure 4.2: Class breakdown of original features experiments

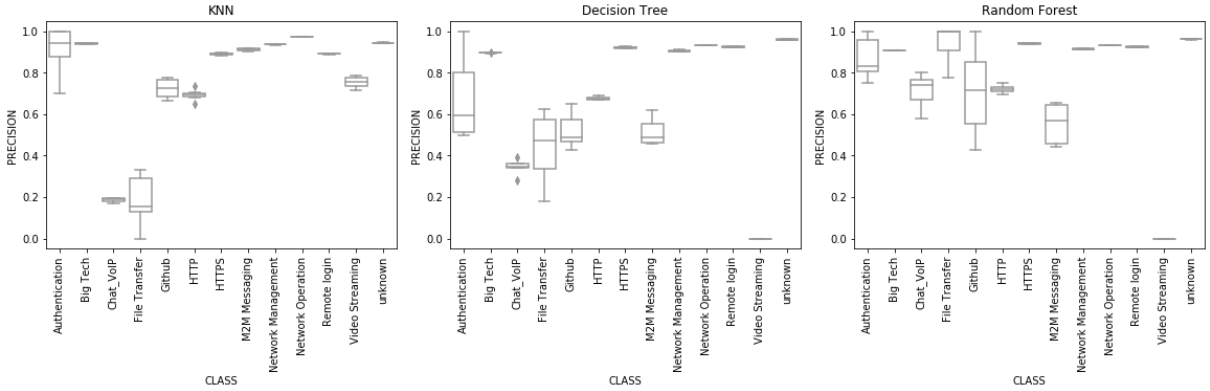


Figure 4.3: Class breakdown of original features with *dstaddrcount* experiments

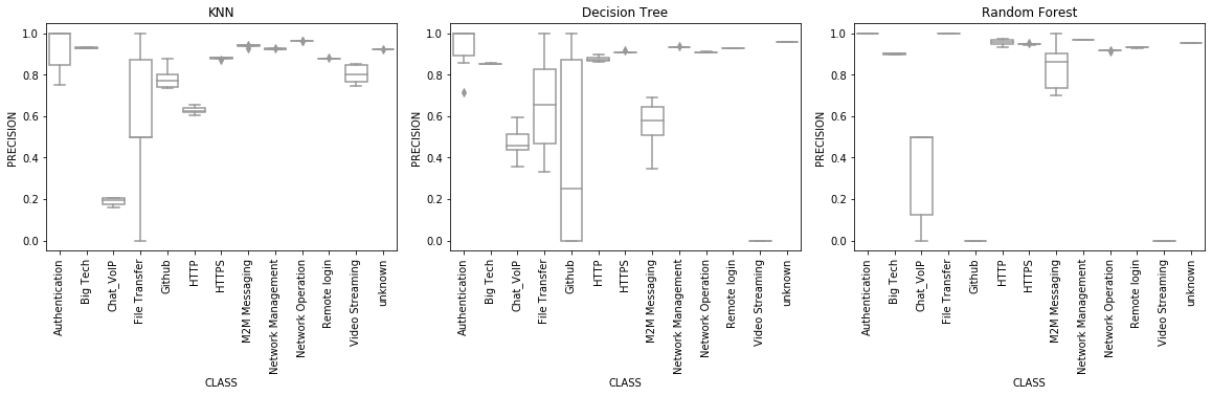


Figure 4.4: Class breakdown of original features with *srcportcount* experiments

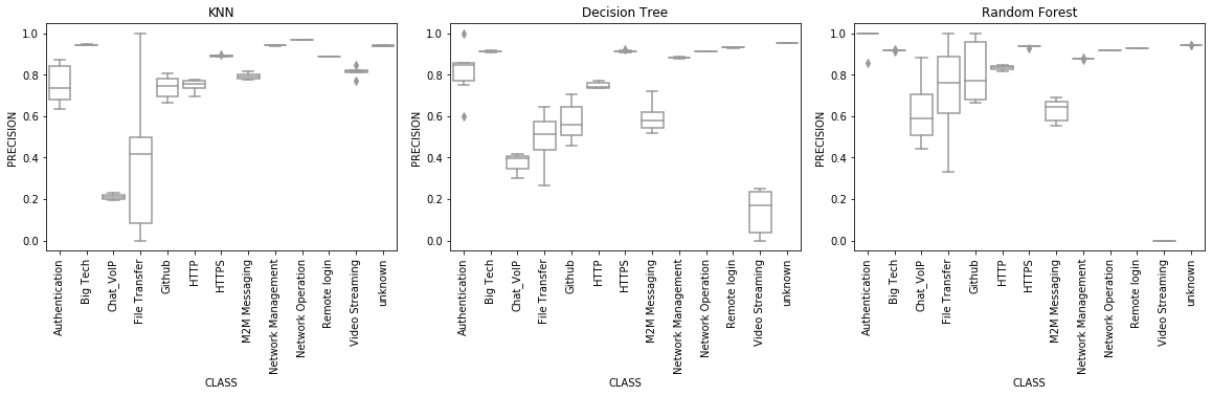


Figure 4.5: Class breakdown of original features with *dstportunique* experiments

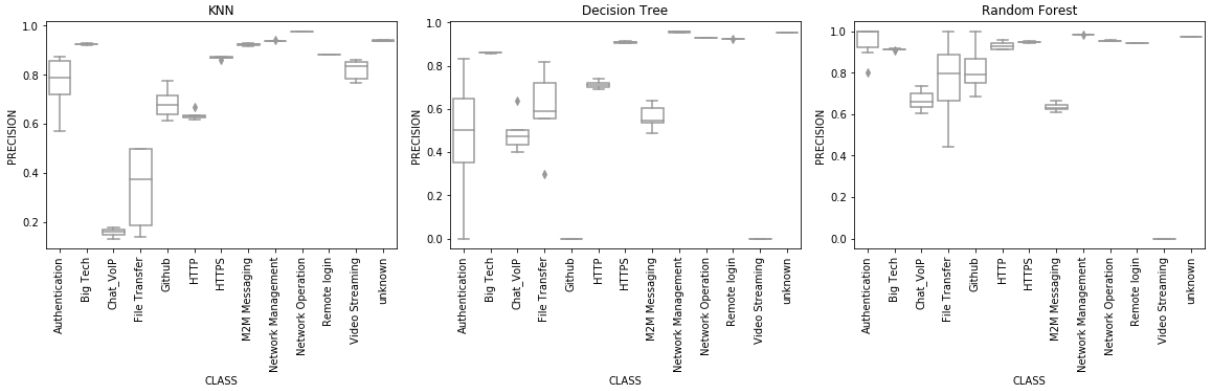


Figure 4.6: Class breakdown of original features with *dstaddrcount* and *srcportcount* experiments

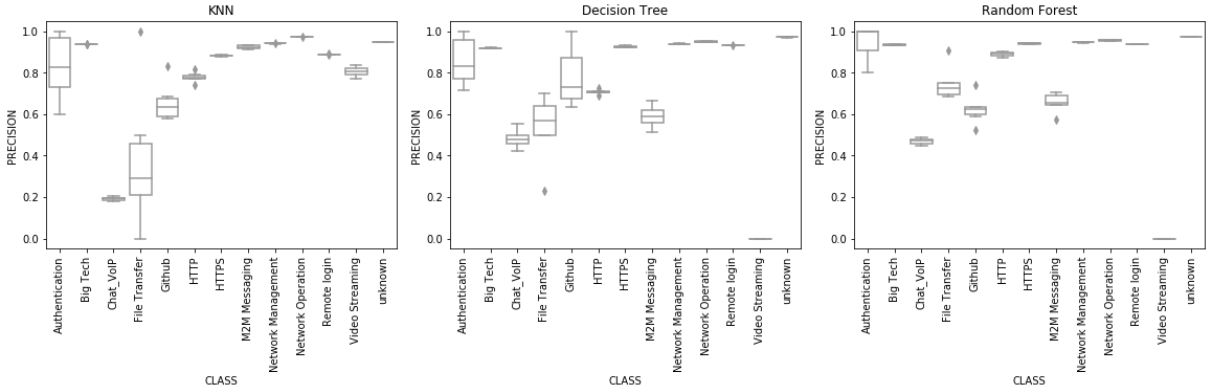


Figure 4.7: Class breakdown of original features with *dstaddrcount* and *dstportunique* experiments

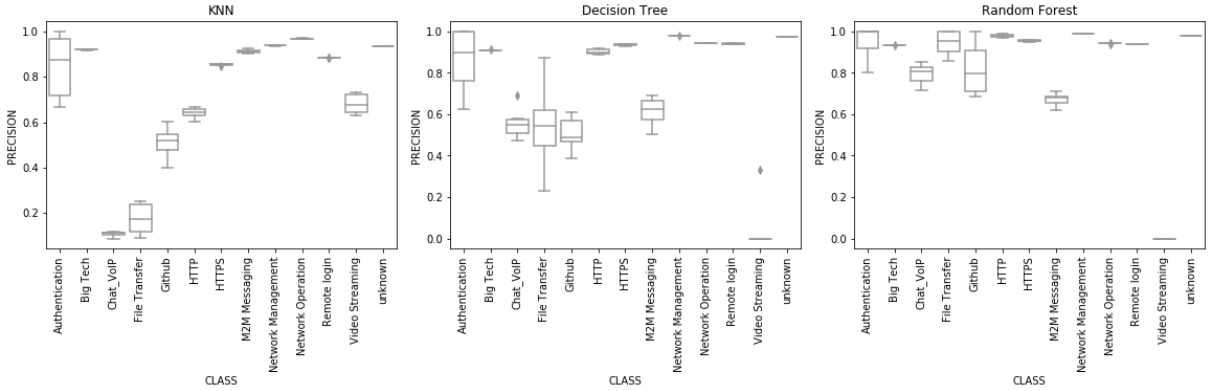


Figure 4.8: Class breakdown of original features with *srcportcount* and *dstportunique* experiments

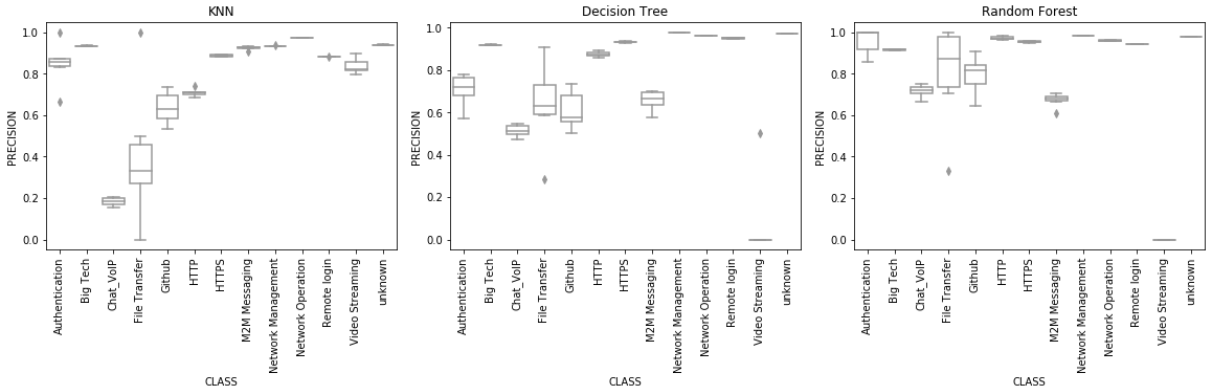


Figure 4.9: Class breakdown of original features with *dstaddrcount*, *srcportcount* and *dstportunique* experiments

Chapter 5

Conclusions

5.1 Experiment Set 1: Can flow features achieve network application classification?

Training data was auto-generated using IANA transport layer port number conventions in a conservative manner. This generated training data is then applied to learn the classification of the network traffic using the three inherent and derived flow features: # of bytes, # of packets, and duration. The experiments show that it is possible to classify certain network applications with reasonable accuracy (close to or above 90%) using the three features. The experiments also revealed that accuracy decreased slightly if the focus is on classifying elephant flows (in our case, the top half of flows with respect to the # of bytes).

5.2 Experiment Set 2: What is the classification performance of combinations of flow features?

Upon examination of the experimental results it can be concluded that:

1. K nearest neighbors had best results when only using the three inherent and derived features (# of packets, # of bytes and duration). Showing that the engineered features degraded the ability to adequately classify. This is likely due to over-fitting because of the increased number of features.

2. Overall, decision trees and random forests presented improvement with the addition

of the engineered behavioral features. This can be attributed to the fact that because they have more data, decision trees and random forests can create more decision nodes which produces a better classification result.

3. *srcportcount* and *dstaddrcount* provide the greatest improvement. This suggests that uncovering the client-server behavior provides insight for network traffic classification.

5.3 Future Work

As more experiments are conducted, several interesting paths to continue the growth of the collective knowledge towards network application classification emerge. In no particular order:

- Diversifying the data set - Both works presented utilized network data captured from an academic institution. Comparing this work across different networks (residential, Internet backbone, commercial, data centers, publicly available data sets) can provide a different perspective which can yield new insights.
- Expanding the feature set - Increasing the number of inherent, derived and engineered flow features that can compliment those analyzed in these efforts provides for an interesting avenue.
- Classification algorithm selection - Comparing a larger set of classification algorithms such as neural networks, semi-supervised learning or creating a framework where a variety of algorithms work in harmony to generate more accurate and robust classifications.

5.4 Final Remarks

These experiments show promise in classifying a section of the network traffic and provide value in confirming that these flow features are adequate for classification of some traffic.

However, as more traffic is included and new applications emerge with the advances in technology, research must also continue to collectively advance the efforts towards improving network application classification and pursuing answers to this complex, but extremely rewarding, problem.

References

[1] "Netflow visualization." http://engineering.utep.edu:62432. Accessed: 2020-04-24.

[2] M. N. Ali, M. Rahman, and S. Hossain, "Network architecture and security issues in campus networks," pp. 1–9, Jul 2013.

[3] "Regional internet registries statistics." https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html. Accessed: 2020-04-26.

[4] "Cisco." https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Medium_Enterprise_Design_Profile/MEDP/chap1.html. Accessed: 2020-04-24.

[5] B. Trammell and B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," 2013.

[6] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 2037–2064, May 2014.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *ACM SIGCOMM computer communication review*, vol. 35, pp. 229–240, ACM, Aug 2005.

[8] "Towards data science." https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0. Accessed: 2020-02-17.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[10] "Datacamp." https://www.datacamp.com/community/tutorials. Accessed: 2019-11-18.

[11] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*, p. 11, ACM, Dec 2008.

[12] "Service name and transport protocol port number registry." https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Accessed: 2019-05-27.

[13] A. C. Callado, C. A. Kamienski, G. Szabó, B. P. Gero, J. Kelner, S. F. Fernandes, and D. F. H. Sadok, "A survey on internet traffic identification.," *IEEE Communications Surveys and Tutorials*, vol. 11, pp. 37–52, Aug 2009.

[14] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE network*, vol. 26, pp. 35–40, Jan 2012.

[15] "ndpi: Open and extensible lgplv3 deep packet inspection library." https://www.ntop.org/products/deep-packet-inspection/ndpi/. Accessed: 2019-06-10.

[16] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "ndpi: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 617–622, IEEE, Sept 2014.

[17] "Pmacct." http://www.pmacct.net. Accessed: 2019-07-22.

[18] P. Lucente, "pmacct: steps forward interface counters," *Tech. Rep.*, 2008.

[19] "Pmacct's official github account." https://github.com/pmacct/pmacct. Accessed: 2019-07-22.

[20] J. Kampeas, A. Cohen, and O. Gurewitz, "Traffic classification based on zero-length packets," *IEEE Transactions on Network and Service Management*, vol. 15, pp. 1049–1062, Apr 2018.

[21] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–5, IEEE, Dec 2016.

[22] C.-C. Liu, Y. Chang, C.-W. Tseng, Y.-T. Yang, M.-S. Lai, and L.-D. Chou, "Svm-based classification mechanism and its application in sdn networks," in *2018 10th International Conference on Communication Software and Networks (ICCSN)*, pp. 45–49, IEEE, Oct 2018.

[23] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2451–2455, IEEE, Oct 2016.

[24] R. Archanaa, V. Athulya, T. Rajasundari, and M. V. K. Kiran, "A comparative performance analysis on network traffic classification using supervised learning algorithms," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1–5, IEEE, Aug 2017.

[25] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 135–148, ACM, Oct 2004.

[26] R. Baker, R. Quinn, J. Phillips, and J. Van der Merwe, "Toward classifying unknown application traffic," in *DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security (DYNAMICS) Workshop*, 2018.

[27] Z. Aouini, A. Kortebi, Y. Ghamri-Doudane, and I. L. Cherif, "Early classification of residential networks traffic using c5. 0 machine learning algorithm," in *2018 Wireless Days (WD)*, pp. 46–53, IEEE, May 2018.

[28] G. Sun, L. Liang, T. Chen, F. Xiao, and F. Lang, "Network traffic classification based on transfer learning," *Computers & electrical engineering*, vol. 69, pp. 920–927, Jul 2018.

[29] X. Bian, "Pso optimized semi-supervised network traffic classification strategy," in *2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 179–182, IEEE, Apr 2018.

[30] P. Wang, S.-C. Lin, and M. Luo, "A framework for qos-aware traffic classification using semi-supervised machine learning in sdns," in *2016 IEEE International Conference on Services Computing (SCC)*, pp. 760–765, IEEE, Sep 2016.

[31] C. Yu, J. Lan, J. Xie, and Y. Hu, "Qos-aware traffic classification architecture using machine learning and deep packet inspection in sdns," *Procedia computer science*, vol. 131, pp. 1209–1216, 2018.

[32] N. Seddigh, B. Nandy, D. Bennett, Y. Ren, S. Dolgikh, C. Zeidler, J. Knoetze, and N. S. Muthyala, "A framework & system for classification of encrypted network traffic using machine learning," in *2019 15th International Conference on Network and Service Management (CNSM)*, pp. 1–5, IEEE, Feb 2020.

[33] F. Noorbehbahani and S. Mansoori, "A new semi-supervised method for network traffic classification based on x-means clustering and label propagation," in *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 120–125, IEEE, Dec 2018.

[34] H.-K. Lim, J.-B. Kim, J.-S. Heo, K. Kim, Y.-G. Hong, and Y.-H. Han, "Packet-based network traffic classification using deep learning," in *2019 International Conference*

on Artificial Intelligence in Information and Communication (ICAIIC), pp. 046–051, IEEE, Mar 2019.

[35] L. Xu, X. Zhou, Y. Ren, and Y. Qin, "A traffic classification method based on packet transport layer payload by ensemble learning," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, January 2020.

[36] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia, "Byte segment neural network for network traffic classification," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Jun 2018.

[37] R. Jin, G. Xue, F. Lyu, H. Sheng, G. Liu, and M. Li, "Leveraging inner-connection of message sequence for traffic classification: A deep learning approach," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 77–84, IEEE, Feb 2018.

[38] Y. Xiao, H. Sun, Z. Zhuang, J. Wang, and Q. Qi, "Common knowledge based transfer learning for traffic classification," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 311–314, IEEE, Feb 2019.

[39] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, and M. J. Deen, "An improved stacked auto-encoder for network traffic flow classification," *IEEE Network*, vol. 32, pp. 22–27, Nov 2018.

[40] T. Li, S. Chen, Z. Yao, X. Chen, and J. Yang, "Semi-supervised network traffic classification using deep generative models," in *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 1282–1288, IEEE, Jul 2018.

[41] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 1271–1276, IEEE, Dec 2017.

[42] V. Tong, H. A. Tran, S. Souihi, and A. Mellouk, "A novel quic traffic classifier based on convolutional neural networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Dec 2018.

[43] "Numfocus: Open code = better science." `https://numfocus.org`. Accessed: 2019-02-19.

[44] "scikit-learn: Machine learning in python." `https://scikit-learn.org/stable/index.html`. Accessed: 2019-02-19.

[45] C. Alcantara, V. Dasari, C. Mendoza, and M. P. McGarry, "Auto-generating training data for network application classification," in *Disruptive Technologies in Information Sciences II*, vol. 11013, p. 1101305, International Society for Optics and Photonics, Apr 2019.

[46] C. Alcantara, V. Dasari, C. Bumgardner, and M. P. McGarry, "Evaluating features for network application classification," in *Disruptive Technologies in Information Sciences IV*, vol. 11419, p. 114190Q, International Society for Optics and Photonics, Apr 2020.

Appendix A

nDPI Label Data Information

In the following appendix, the complete list of nDPI labels found in the UKY data set are reported along with flow counts and the definition of the application label.

A.1 Label Definitions

Table A.1: nDPI application label definitions.

App Label	Count	Description
Unknown	3480677	nDPI is unable to classify this flow
SSH	1995177	Also referred to as Secure Shell, is a method for secure remote login from one computer to another.
ICMP	1005147	(Internet Control Message Protocol) is an error-reporting protocol that network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets. ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet, i.e. a router, service or host cannot be reached for packet delivery. Any IP network device has the capability to send, receive or process ICMP messages.

Table A.1 continued

App Label	Count	Description
DNS	933846	Hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Typically used to translate more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.
SNMP	525298	Simple Network Management Protocol (SNMP) is an Internet Standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior. Devices that typically support SNMP include cable modems, routers, switches, servers, workstations, printers, and more. SNMP is widely used in network management for network monitoring. SNMP exposes management data in the form of variables on the managed systems organized in a management information base (MIB) which describe the system status and configuration. These variables can then be remotely queried (and, in some circumstances, manipulated) by managing applications.
NTP	257972	The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems. NTP is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC).

Table A.1 continued

App Label	Count	Description
Google	202861	Internet-related services and products, which include online advertising technologies, search engine, cloud computing, software, and hardware. SSL Certificate verified as belonging to Google.
SSL_No_Cert	110155	HTTP with SSL/TLS security protocol but no certificate found in payload. This happens because by default only the first few (7 or 8) packet payloads of each flow are analyzed.
UbuntuONE	40659	Free suite of cloud services that provides users with online cloud storage, syncing, sharing and streaming capabilities for managing personal data across numerous devices operating on a variety of operating systems.
Syslog	27542	Standard for message logging. Uses a client-server architecture where the server listens on a well-known or registered port for protocol requests from clients. Historically the most common transport layer protocol for network logging has been User Datagram Protocol (UDP), with the server listening on port 514. As UDP lacks congestion control mechanisms, support for Transport Layer Security is required in implementations and recommended for general use on Transmission Control Protocol (TCP) port 6514.

Table A.1 continued

App Label	Count	Description
ICMPV6	23533	Internet Control Message Protocol version 6 (ICMPv6) is the implementation of the Internet Control Message Protocol (ICMP) for Internet Protocol version 6 (IPv6).,ICMPv6 is an integral part of IPv6 and performs error reporting and diagnostic functions (e.g., ping), and has a framework for extensions to implement future changes.
STUN	18213	Session Traversal Utilities for NAT (STUN) is a standardized set of methods, including a network protocol, for traversal of network address translator (NAT) gateways in applications of real-time voice, video, messaging, and other interactive communications. STUN is a tool used by other protocols, such as Interactive Connectivity Establishment (ICE), the Session Initiation Protocol (SIP), or WebRTC. It provides a tool for hosts to discover the presence of a network address translator, and to discover the mapped, usually public, Internet Protocol (IP) address and port number that the NAT has allocated for the application's User Datagram Protocol (UDP) flows to remote hosts. The protocol requires assistance from a third-party network server (STUN server) located on the opposing (public) side of the NAT, usually the public Internet.
GoogleServices	18202	Background service that runs on Android, which helps in integrating Google's advanced functionalities to other applications.

Table A.1 continued

App Label	Count	Description
SSDP	16829	Simple Service Discovery Protocol (SSDP) is a network protocol based on the Internet protocol suite for advertisement and discovery of network services and presence information. It accomplishes this without assistance of server-based configuration mechanisms, such as Dynamic Host Configuration Protocol (DHCP) or Domain Name System (DNS), and without special static configuration of a network host. SSDP is the basis of the discovery protocol of Universal Plug and Play (UPnP) and is intended for use in residential or small office environments. It was formally described in an Internet Engineering Task Force (IETF) Internet Draft by Microsoft and Hewlett-Packard in 1999. Although the IETF proposal has since expired (April, 2000), SSDP was incorporated into the UPnP protocol stack, and a description of the final implementation is included in UPnP standards documents.
Amazon	14255	Amazon and Amazon Data Services. SSL Certificate verified.
HTTP	10991	The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access by a mouse click or by tapping the screen in a web browser.

Table A.1 continued

App Label	Count	Description
QQ	3505	Tencent QQ, generally referred to as QQ, is the most popular free instant messaging computer program in China.
Viber	3482	Cross-platform voice over IP (VoIP) and instant messaging (IM) software application operated by Japanese multinational company Rakuten, provided as a freeware for the Android, iOS, Microsoft Windows, macOS and Linux platforms.
DHCPV6	2985	Dynamic Host Configuration Protocol version 6 (DHCPv6) is a network protocol for configuring Internet Protocol version 6 (IPv6) hosts with IP addresses, IP prefixes and other configuration data required to operate in an IPv6 network. It is the IPv6 equivalent of the Dynamic Host Configuration Protocol,(DHCP)for IPv4.

Table A.1 continued

App Label	Count	Description
QUIC	1445	General-purpose transport layer network protocol initially designed by Jim Roskind at Google. Although its name was initially proposed as the acronym for "Quick UDP Internet Connections", IETF's use of the word QUIC is not an acronym; it is simply the name of the protocol.,Among other applications, QUIC improves performance of connection-oriented web applications that are currently using TCP. It does this by establishing a number of multiplexed connections between two endpoints over User Datagram Protocol (UDP). This works hand-in-hand with HTTP/2's multiplexed connections, allowing multiple streams of data to reach all the endpoints independently, and hence independent of packet losses involving other streams. In contrast, HTTP/2 hosted on Transmission Control Protocol (TCP) can suffer head-of-line-blocking delays of all multiplexed streams if any of the TCP packets are delayed or lost.,QUIC's secondary goals include reduced connection and transport latency, and bandwidth estimation in each direction to avoid congestion. It also moves congestion control algorithms into the user space at both endpoints, rather than the kernel space, which it is claimed will allow these algorithms to improve more rapidly. Additionally, the protocol can be extended with forward error correction (FEC) to further improve performance when errors are expected. QUIC is often used by gaming, streaming media and VoIP services.

Table A.1 continued

App Label	Count	Description
YouTube	1347	Video-sharing platform. SSL Certificate verified.
NetBIOS	1240	OSI Session Layer 5 Protocol and a service that allows applications on computers to communicate with one another over a local area network (LAN). It is a non-routable Protocol and NetBIOS stands for Network Basic Input/Output System.
Apple	1038	Company that designs, develops, and sells consumer electronics, computer software, and online services. SSL Certificate verified.
Github	719	Provides hosting for software development version control using Git. SSL Certificate verified.
ApplePush	394	Apple Push Notification service (commonly referred to as Apple Notification Service or APNs) is a platform notification service created by Apple Inc. that enables third party application developers to send notification data to applications installed on Apple devices.
RDP	389	Remote Desktop Protocol (RDP) is a proprietary protocol developed by Microsoft, which provides a user with a graphical interface to connect to another computer over a network connection. The user employs RDP client software for this purpose, while the other computer must run RDP server software.

Table A.1 continued

App Label	Count	Description
Cloudflare	371	Cloudflare, Inc. is an American web infrastructure and website security company, providing content delivery network services, DDoS mitigation, Internet security, and distributed domain name server services. Cloudflare's services sit between a website's visitor and the Cloudflare user's hosting provider, acting as a reverse proxy for websites.
FTP_CONTROL	341	The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
SkypeCall	297	The Skype protocol is a proprietary Internet telephony network used by Skype. The protocol's specifications have not been made publicly available by Skype and official applications using the protocol are closed-source.
Facebook	201	Social media and networking service traffic over HTTP. SSL Certificate verified.

Table A.1 continued

App Label	Count	Description
Kerberos	176	Computer-network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Its designers aimed it primarily at a clientserver model and it provides mutual authentication-both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks. Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication. Kerberos uses UDP port 88 by default.
Tor	90	Tor is free and open-source software for enabling anonymous communication. The name is derived from an acronym for the original software project name "The Onion Router". Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis.

Table A.1 continued

App Label	Count	Description
SSL	56	Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers. Traffic receives this label when a SSL/TLS certificate is found in the payload but the certificate does not match any of the known certificates thus no subprotocol is declared.
IGMP	49	The Internet Group Management Protocol (IGMP) is a communications protocol used by hosts and adjacent routers on IPv4 networks to establish multicast group memberships. IGMP is an integral part of IP multicast. IGMP can be used for one-to-many networking applications such as online streaming video and gaming, and allows more efficient use of resources when supporting these types of applications.

Table A.1 continued

App Label	Count	Description
MQTT	49	A machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers.
TeamSpeak	45	Proprietary voice-over-Internet Protocol (VoIP) application for audio communication between users on a chat channel, much like a telephone conference call. The client software connects to a TeamSpeak server of the user's choice, from which the user may join chat channels.
RTMP	44	Real-Time Messaging Protocol (RTMP) was initially a proprietary protocol developed by Macromedia for streaming audio, video and data over the Internet, between a Flash player and a server. Macromedia is now owned by Adobe, which has released an incomplete version of the specification of the protocol for public use. While the primary motivation for RTMP was to be a protocol for playing Flash video, it is also used in some other applications, such as the Adobe LiveCycle Data Services ES.

Table A.1 continued

App Label	Count	Description
GoogleHangout	28	Google Hangouts is a communication software product developed by Google. Google began developing Hangouts into a product aimed at enterprise communication. Hangouts is now part of the G Suite line of products and consists of two primary products: Google Hangouts Meet and Google Hangouts Chat. Google has also begun integrating features of Google Voice, its IP telephony product, into Hangouts, stating that Hangouts is designed to be "the future" of Voice.
Twitter	20	Social networking service. SSL Certificate verified.
PlayStore	19	Google Play is a digital distribution service operated and developed by Google. It serves as the official app store for the Android operating system, allowing users to browse and download applications developed with the Android software development kit and published through Google.
GMail	19	Free email service developed by Google.
MS_OneDrive	18	File hosting service and synchronization service operated by Microsoft as part of its web version of Office.

Table A.1 continued

App Label	Count	Description
Whois-DAS	18	The Whois service provides a way for the public to lookup information about registered data. To protect the data from possible abuse, the Whois service enforces a rate limitation mechanism, which will limit the possibility for a client to do a large number of requests within a short period of time. DAS (Domain Availabilty Service) provides a way for the public to check whether a domain can be registered. The DAS service enforces a lower rate limitation than the Whois.
Office365	16	Cloud-based services offered by Microsoft as part of the Microsoft Office product line.
GoogleDrive	12	File storage and synchronization service developed by Google which allows users to store files on their servers, synchronize files across devices, and share files. In addition to a website, Google Drive offers apps with offline capabilities for Windows and macOS computers, and Android and iOS smartphones and tablets.
Skype	12	Telecommunications application that specializes in providing video chat and voice calls between computers, tablets, mobile devices, the Xbox One console, and smartwatches via the Internet. Skype also provides instant messaging services. Users may transmit text, video, audio and images.

Table A.1 continued

App Label	Count	Description
Microsoft	8	Technology company that develops, manufactures, licenses, supports, and sells computer software, consumer electronics, personal computers, and related services. SSL Certificate verified.
COAP	8	Constrained Application Protocol (CoAP) is a specialized Internet Application Protocol for constrained devices, as defined in RFC 7252. It enables those constrained devices called "nodes" to communicate with the wider Internet using similar protocols. CoAP is designed for use between devices on the same constrained network (e.g., low-power, lossy networks), between devices and general nodes on the Internet, and between devices on different constrained networks both joined by an internet. CoAP is also being used via other mechanisms, such as SMS on mobile communication networks. CoAP is a service layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensor network nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity. Multicast, low overhead, and simplicity are extremely important for Internet of Things (IoT) and Machine-to-Machine (M2M) devices, which tend to be deeply embedded and have much less memory and power supply than traditional internet devices have. Therefore, efficiency is very important. CoAP can run on most devices that support UDP.

Table A.1 continued

App Label	Count	Description
IRC	8	Internet Relay Chat (IRC) is an application layer protocol that facilitates communication in the form of text. The chat process works on a client/server networking model. IRC is mainly designed for group communication in discussion forums, called channels, but also allows one-on-one communication via private messages as well as chat and data transfer, including file sharing.
Redis	6	Redis (Remote Dictionary Server) is an in-memory data structure project implementing a distributed, in-memory key-value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes. Used as a database, a caching layer or a message broker.
Oscar	6	OSCAR (Open System for CommunicAtion in Realtime) is AOL's proprietary instant messaging and presence information protocol. It was used by AOL's AIM instant messaging system and ICQ, their VoIP which name comes from the phrase "I Seek You".
BitTorrent	5	A communication protocol for peer-to-peer file sharing (P2P) which is used to distribute data and electronic files over the Internet. BitTorrent is one of the most common protocols for transferring large files, such as digital video files containing TV shows or video clips or digital audio files containing songs.

Table A.1 continued

App Label	Count	Description
AppleiCloud	4	iCloud is a cloud storage and cloud computing service from Apple Inc. SSL Certificate verified.
NFS	4	Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems (Sun) in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system.
LinkedIn	4	Social media platform tailored to professionals. SSL Certificate verified.
Teredo	3	Transition technology that gives full IPv6 connectivity for IPv6-capable hosts that are on the IPv4 Internet but have no native connection to an IPv6 network. Unlike similar protocols such as 6to4, it can perform its function even from behind network address translation (NAT) devices such as home routers. Teredo operates using a platform independent tunneling protocol that provides IPv6 (Internet Protocol version 6) connectivity by encapsulating IPv6 datagram packets within IPv4 User Datagram Protocol (UDP) packets. Teredo routes these datagrams on the IPv4 Internet and through NAT devices.

Table A.1 continued

App Label	Count	Description
MDNS	3	Multicast DNS (mDNS) protocol resolves hostnames to IP addresses within small networks that do not include a local name server. It is a zero-configuration service, using essentially the same programming interfaces, packet formats and operating semantics as the unicast Domain Name System (DNS). Although Stuart Cheshire designed mDNS as a stand-alone protocol, it can work in concert with standard DNS servers.
BGP	2	Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet.
GoogleMaps	2	Web mapping service developed by Google. SSL Certificate verified.
GoogleDocs	2	Word processor included as part of a free, web-based software office suite offered by Google within its Google Drive service. This service also includes Google Sheets and Google Slides, a spreadsheet and presentation program respectively. Google Docs is available as a web application, mobile app for Android, iOS, Windows, BlackBerry, and as a desktop application on Google's ChromeOS.

Table A.1 continued

App Label	Count	Description
UPnP	2	Universal Plug and Play (UPnP) is a set of networking protocols that permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment. UPnP is intended primarily for residential networks without enterprise-class devices.
FTP_DATA	2	The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
SIP	1	The Session Initiation Protocol (SIP) is a signaling protocol used for initiating, maintaining, and terminating real-time sessions that include voice, video and messaging applications. SIP is used for signaling and controlling multimedia communication sessions in applications of Internet telephony for voice and video calls, in private IP telephone systems, in instant messaging over Internet Protocol (IP) networks as well as mobile phone calling over LTE (VoLTE).

A.2 Flow Data Breakdown by Label

Table A.2: Flow counts by nDPI application label.

App	Count	Percent
Total	8699852	100.0000
Unknown	3480677	40.0085
SSH	1995177	22.9335
ICMP	1005147	11.5536
DNS	933846	10.7340
SNMP	525298	6.0380
NTP	257972	2.9652
Google	202861	2.3318
SSL_No_Cert	110155	1.2662
UbuntuONE	40659	0.4674
Syslog	27542	0.3166
ICMPV6	23533	0.2705
STUN	18213	0.2093
GoogleServices	18202	0.2092
SSDP	16829	0.1934
Amazon	14255	0.1639
HTTP	10991	0.1263
QQ	3505	0.0403
Viber	3482	0.0400
DHCPV6	2985	0.0343
QUIC	1445	0.0166
YouTube	1347	0.0155

App	Count	Percent
NetBIOS	1240	0.0143
Apple	1038	0.0119
Github	719	0.0083
ApplePush	394	0.0045
RDP	389	0.0045
Cloudflare	371	0.0043
FTP_CONTROL	341	0.0039
SkypeCall	297	0.0034
Facebook	201	0.0023
Kerberos	176	0.0020
Tor	90	0.0010
SSL	56	0.0006
IGMP	49	0.0006
MQTT	49	0.0006
TeamSpeak	45	0.0005
RTMP	44	0.0005
GoogleHangout	28	0.0003
Twitter	20	0.0002
PlayStore	19	0.0002
GMail	19	0.0002
MS_OneDrive	18	0.0002
Whois-DAS	18	0.0002
Office365	16	0.0002
GoogleDrive	12	0.0001
Skype	12	0.0001
Microsoft	8	0.0001

App	Count	Percent
COAP	8	0.0001
IRC	8	0.0001
Redis	6	0.0001
Oscar	6	0.0001
BitTorrent	5	0.0001
AppleiCloud	4	0.0000
NFS	4	0.0000
LinkedIn	4	0.0000
Teredo	3	0.0000
MDNS	3	0.0000
BGP	2	0.0000
GoogleMaps	2	0.0000
GoogleDocs	2	0.0000
UPnP	2	0.0000
FTP_DATA	2	0.0000
SIP	1	0.0000

Curriculum Vitae

Carlos Alcantara completed his Bachelor of Science in Electrical and Computer Engineering at The University of Texas at El Paso (UTEP) in December 2018, where he continued his education by pursuing a Master's of Science in Computer Engineering under the supervision of Dr. Michael P. McGarry. Carlos is a Graduate Research Assistant in the UTEP Communication Networks Laboratory. His research interests are in the application of data analysis, artificial neural networks and machine learning in communication networks.

Carlos' work has been recognized in two separate publications, the first "Auto-generating training data for network application classification" was published on April of 2019. His second publication, "Evaluating features for network application classification," was published on April 2020. Carlos is noted as the first author in both of these works.

Email: calcantara@miners.utep.edu