Toward Classifying Unknown Application Traffic

Ryan Baker
University of Utah
ryan.baker@utah.edu

Ren Quinn
University of Utah
renquinn@cs.utah.edu

Jeff Phillips
University of Utah
jeffp@cs.utah.edu

Jacobus van der Merwe
University of Utah
kobus@cs.utah.edu

ABSTRACT

Determining the particular application associated with a given flow of internet traffic is an important security measure in computer networks. This practice is significant as it can aid in detecting intrusions and other anomalies, as well as identifying misuse associated with prohibited applications. Many efforts have been expended to create models for classifying internet traffic using machine learning techniques. While research so far has proven useful, studies have focused on machine learning techniques for detecting well-known and profiled applications. Some have focused only on particular transport layer traffic (e.g., TCP traffic only). In contrast, unknown traffic is much more difficult to classify and can appear as previously unseen applications or established applications exhibiting abnormal behavior. This work presents methods to address these gaps in other research. The methods utilize k-Nearest Neighbor machine learning approaches to model known application data with the Kolmogorov-Smirnov statistic as the distance function to computer nearest neighbors. The models identify incoming data which likely does not belong to the model, thus identifying unknown applications. This study shows the potential of our approach by presenting results which show successful implementation for a controlled environment, such as an organization with a fixed number of approved applications. In this setting, our approach can distinguish unknown data from known data with accuracy up to 93 percent compared to an accuracy of 57 percent for a strawman k-Nearest Neighbors approach with Euclidean distance. In addition, there are no restrictions on particular protocols. Operational considerations are also discussed, with emphasis on future work that can be performed such as exploring processing of incoming data in real-time and updating the model in an automated way.

ACM Reference format:
Ryan Baker, Ren Quinn, Jeff Phillips, and Jacobus van der Merwe. 2018. Toward Classifying Unknown Application Traffic. In *Proceedings of DYNAMICS'18, San Juan, Puerto Rico, USA, December 2018,* 8 pages.

1 INTRODUCTION

In computer networking, large amounts of traffic flow through individual networks at very high rates. A huge amount of data can be gathered and this is typically well beyond the scope of a network manager to evaluate. This is particularly true for tasks that require close to real-time evaluation, such as network security pursuits. One important task of this type is traffic application classification, which identifies the application(s) and protocol(s) associated with a given traffic flow. This is an important pursuit because it allows malicious or inappropriate application use in a network to be identified and handled. However, an often overlooked, yet critically important security consideration for application identification is that there are constantly new types of applications released for consumer use, some of which may be malicious or inappropriate. These are sometimes referred to as 0-day applications [30]. Furthermore, many applications will be disguised in unique ways because of updates to an application or explicit efforts to prevent discovery, such that the traffic looks unlike any other application. Such applications are sometimes referred to as obfuscated or dynamic applications [9]. These are examples of unknown application traffic that a security system will be exposed to amidst other well-known and profiled applications. Operational systems have proven to be quite robust against well-known applications, but suffer against unknown applications [22]. Additionally, a significant percentage of operational network traffic is in reality unknown application traffic [20]. This is a major vulnerability that can leave networks exposed to dangerous applications for significant time periods before detection.

The focus of this work is to address the problem of unknown application classification. While some work has been completed on classifying unknown application traffic, previous efforts have been encumbered by strict constraints (e.g., particular transport layer protocols or deep packet inspection), and the general problem is considered an open challenge [9, 14, 19, 22, 36]. This work seeks to contribute to gaps in the current research landscape by implementing a machine learning approach to classify unknown application traffic. This is done without imposing restrictions of any kind on the traffic that can be processed.

Ideally, a machine learning approach will be able to identify unknown traffic within a large and complex network, such as a full university or corporate network. In these types of networks, it would be desireable to identify unknown traffic specifically to prevent malicious or inappropriate activity such as the spread of malware or the use of Netflix at work. In these settings, great care is required to create an accurate machine learning approach to distinguish between applications which are truly malicious and/or inappropriate and applications which are benign, but quite similar to other bad applications. In particular, it requires thorough knowledge of thousands of different applications and the ability to devise a feature set which can describe differences between many different types of allowable traffic and many different types of bad traffic.

However, this study focuses on a more specific environment, namely, the approach presented constrains the problem to smaller, more controlled networks. Examples of these could be a financial institution or hospital with a specific set of approved applications which can be utilized on the network. Then the goal in these circumstances is to identify any use of the network which is not one of the approved applications, whether or not it is explicitly malicious or inappropriate. So, it is realistic and appropriate in these circumstances to first define a narrow set of applications. With a thorough knowledge of this limited set of applications, it is much more feasible to devise a feature set which accurately and very narrowly can separate these applications from any other applications. For example, in a bank with only one specific application used for transactions and customer service, the specific purpose of the machine learning model includes separating out general web-browsing as well as a piece of malware. Even though the general web-browsing isn't explicitly malicious or inappropriate in any way, it is realistic in this setting to filter out both. Furthermore, when building a machine learning model with a controlled set of data from the list of approved applications, constraining the approach to focus on these specific environments is operationally viable because a successful approach will successfully monitor any application traffic which does not belong to this approved list, including any and all malicious traffic. Critically, this also provides a more controlled setting to begin exploring the use of machine learning algorithms to identify unknown traffic in networks generally. This is done by treating all application traffic which does not fit within the approved list of applications as unknown, regardless of whether or not it is actually unknown within a larger, unrestricted network. This approach is also specific and pragmatic, a feature which is rarely found in previously published research on machine learning application classification [17].

The machine learning approach developed in this research uses a k-Nearest Neighbors classifier that requires an initial set of explicitly labeled data. A critical component of the model is the selection of the most appropriate distance function, in this case the Kolmogorov-Smirnov statistic. To identify unknown application traffic using the k-Nearest Neighbors model, the distances between a new point and its closest neighbors are compared with the distances between the new point's neighbors and their closest neighbors. The key to this comparison is using a hyper-parameter threshold, which is tuned using cross-validation, to specify how much "further" from its neighbors a new point needs to be to be accurately classified as an unknown application, rather than just an outlier for a known application. With this approach, the classifier is able to achieve an overall accuracy of 92.67 %, with 91.98 % accuracy on known traffic and 92.81 % accuracy on unknown traffic.

Moreover, internet traffic classification is an invaluable tool for network operators to identify network misuse, such as intrusions from outside the network and violations of the network usage policies established by a university, company, or other organization. This work will significantly aid network operators in maintaining a secure network environment by enabling real-time identifications of unknown traffic that were previously very difficult or even impossible.

The main contributions of this work are as follows:

(1) Gathering specific application data and extracting a set of features that can be used to distinguish traffic from other applications.
(2) Showing that a machine learning approach with a k-nearest-neighbors model, used with the Kolmogorov-Smirnov statistic, can accurately identify traffic from unknown applications, without restrictions on traffic type (e.g., a specific network protocol).
(3) Discussing future research directions for utilizing this approach in an operational, and possibly a more general, system.

The next section of this paper discusses the background and related work of this problem. The following few sections discuss the architecture, data and methodologies. Finally, the paper concludes with an evaluation and discussion of operational implementation and future work.

2 RELATED WORK

Machine learning techniques have been used heavily in network management and security settings to analyze large amounts of data. These methods include supervised methods, which require explicit labels for data points, and unsupervised methods, which do not require any labels. These methods have been used in a number of papers for application classification tasks [6, 12, 13, 16, 18, 21, 27, 32, 34] . Many of these methods show great potential and have been successful in achieving classification accuracy of up to 99%. Another group of approaches uses semi-supervised learning, a blend of these two styles, to produce robust and flexible models [38]. Some studies focus on very specific protocols and traffic types, such as HTTPS services [28]. Other studies focus on tackling specific sub-problems of classification such as high accuracy in cross-platform scenarios [15].

However, creating models to address general classification of unknown traffic is still an open research challenge. A small set of studies has been able to achieve very high accuracy for unknown traffic classification, but is dependent on deep packet inspection (DPI) [29, 30]. Unfortunately, DPI has significant impact in an operational system in order to examine the payload contents of a network packet. Not only is this process expensive and slow, but it is encumbered by strict security and privacy issues. Another study is able to obtain 60% accuracy for identifying unknown application traffic, but maintains constraints that only TCP traffic can be considered, which eliminates a huge amount of other traffic (e.g., UDP) [6]. The researchers utilize labeled data, which is difficult or perhaps impossible to obtain in a real-time evaluation setting, but are able to achieve their results by only looking at the first few packets of a connection in the network. Further still, other researchers have combined these previous two types of schemes that can selectively filter packets that should be examined with DPI, without sacrificing accuracy [23, 25]. There has also been work completed for classifying new applications, but is specifically based upon bot detection

[14]. Some work has shown promise using unsupervised methods, which don't explicitly require labels, but these works also don't study unknown traffic situations [12, 13].

There are also many corporate systems that incorporate machine learning into network security products. In conjunction with many of these systems, companies have published white papers on their use of machine learning in network security products. However, these systems are proprietary and white papers do not include any details about the actual implementation or results of machine learning methodology in the systems. LogRhythm [3], IBM [1], and Juniper [2] are among those who have published such white papers.

Besides the network setting and focus of a study, the machine learning modeling choices are also of great significance. These modeling choices incorporate both algorithmic and statistical factors. While tackling application classification, researchers have used diverse categories of machine learning algorithms, such as clustering and probabalistic algorithms. Examples of these include k-Nearest-Neighbor models [10] and Gaussian mixture models [31], respectively. There are also statistical metrics that must be selected to use in conjuction with each algorithm. Some examples of these statistics are the Kolmogorov-Smirnov statistic [24] and the Kullback-Leibler divergence [18]. Each unique data set will respond differently to a particular machine learning model, and the data is factored heavily into these choices. The methodology for this study is based upon k-Nearest-Neighbors clustering combined with the Kolmogorov-Smirnov statistic, and these choices are qualified in subsequent sections.

Other work has also been useful to review to conduct this research using machine learning. One of the most important aspects of machine learning approaches is the ability to extract useful features from a data set to use in a machine learning model. In fact, the feature selection associated with a data set can impact results as much or more than the choice of machine learning model [5, 35]. Netflow data is a particularly portable and convenient set of data to obtain that describes traffic in a network. While other more robust data can be obtained, it is more expensive and intrusive, which again has potentially dooming privacy implications with respect to operational settings. Using netflow data in machine learning models for application classification is the current focus of most research because of the ease associated with its collection [19]. On the other hand, netflow data is less descriptive and it is difficult to classify dynamic applications [19] and general (including unknown) applications [36].

Many authors detail how traffic flow data may be analyzed to form different features that are used in machine learning algorithms [8, 11, 16, 21, 26, 32–35, 37]. Arzani et al. [7] describe in detail some of the features they use for TCP related work. Examples from their work include maximum congestion windows, changes in congestion windows, round trip time (RTT) estimates, number of duplicate ACKs, duration of the connection, and the number of bytes sent and received. This previous work in feature selection is instrumental in identifying a starting point for extracting useful features in this study. To fully utilize flow statistics as features, there has also been work indicating the effectiveness of using 5-tuple based connections as distinct entities for gathering statistics [31]. Similarly, the data gathered and used for the work presented here will also incorporate these results in an effort to build off previous successes. This is described in coming sections. Finally, the machine learning tools available for these types of endeavors are important to acknowledge. SciKit-Learn is an important package available in Python that provides access to many machine learning techniques and tools, and is used extensively in this project [4].

3 ARCHITECTURE

3.1 Network Management Setting and Threat Model

This research focuses on a particular management setting. In order to evaluate traffic in real-time, network management staff needs tools and devices for continuous monitoring of traffic. At one or more edge points, firewalls or other devices can continuously collect network data. This network data may contain various levels of flow details, but typically contain details such as IP addresses, port numbers, and packet and byte counts [19]. In addition to the device(s) collecting network data, the network management staff maintains a database of network traffic data. It is assumed that data in this database contain known application information. At any given time, the database can be queried for data, which can be used to form feature vectors that are then used to create machine learning models. As new traffic comes in and goes out, network management staff have the task of identifying inappropriate traffic. The similar edge device, possibly the same device, can create feature vectors from incoming traffic and use machine learning models to classify whether or not this traffic is inappropriate. When an edge device determines that a batch of inbound or outbound traffic doesn't match known traffic, the edge device can notify the network management staff for further evaluation. In an operational environment it is important for traffic to be evaluated right away and both known and unknown traffic is collected. In a controlled environment, used in this study to develop and evaluate a methodology and in general to create an initial model, only known traffic is collected and then stored. The stored data is then used to develop a model. In this study, the controlled data is also used to evaluate the methodology. In particular, a portion of the controlled, known traffic is simulated as unknown traffic to evaluate the model as if it were operational (Figure 1).

This research focuses on two different forms of misuse as a threat model. The first form is intrusive and often malicious misuse, largely characterized by unsolicited or undesired network traffic. For a constrained use case such as a bank, examples of this form of misuse include malware, worms, and viruses intended to compromise sensitive information and financial security. The second form is native misuse that originates from sources inside the network. This form of misuse is often characterized by intentional activity that violates network acceptable use policies. Again considering the use case of constrained bank network, an example of this might be gaming on the network. For both of these forms of misuse, the threat model for this study assumes that traffic crosses a network edge boundary. That is, the traffic either originates in the network and is sent outside the network, or the traffic originates outside the network and is sent into the network.

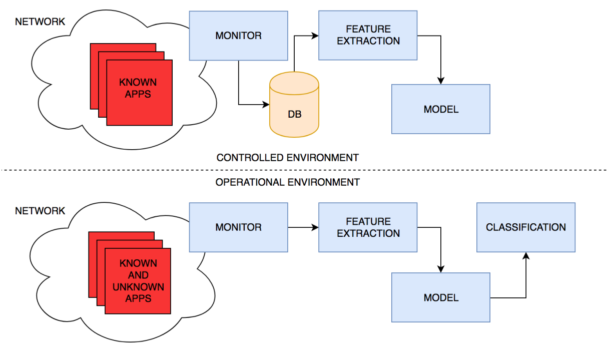

Figure 1: Setting - Monitoring inbound/outbound traffic for identification of unknown applications

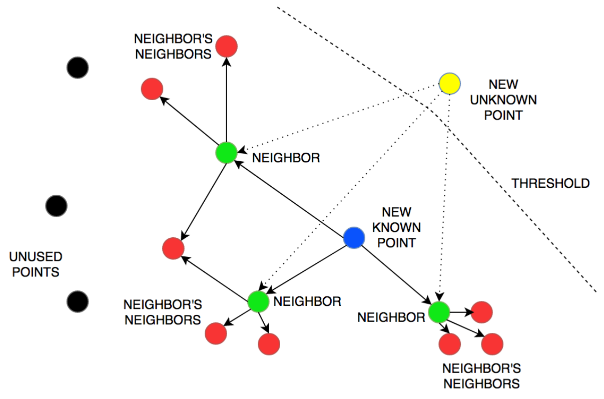

Figure 2: kNN Model Comparisons - Distance between new point, its neighbors, and its neighbor's neighbors

3.2 Problem Formulation

As described in the network setting, a device may be placed at the edge of a network to collect network traffic data, and known application data is stored in a database. To form a basis for comparison, this data is used to create a model using a machine learning algorithm. In this case, the unsupervised machine learning method used is a k-Nearest Neighbors clustering algorithm. Once a base model has been formed, it is possible to begin analyzing incoming data. The edge device will record network traffic data and form a "batch" of incoming data. The exact methods for separating masses of incoming traffic into individual batches is left to future work. One example, depending on the needs of a network, is that application traffic might be grouped based upon the IP address of the client within the network being monitored. Batches of incoming data will be processed into individual data points with the appropriate feature values.

Recall that the k-Nearest Neighbors model is trained with only known, or accepted, application traffic. Network management staff are interested in finding traffic which is unknown, or is not consistent with known application traffic. In contrast to traditional k-Nearest Neighbors, which identifies the best classification for a new point based upon its neighbors, this method determines whether a new data point is sufficiently far from all other neighbors that the classifier should predict that the new point is unknown relative to the application traffic used to build the model.

To accomplish this, the model computes the average distance from the new data point (a new application session) to its k nearest neighbors. Let this new data point be x, and let each neighbor be y_i for $1 \le i \le k$. Then, for each of these k nearest neighbors, the model computes the average distance to their k nearest neighbors. Let the neighbors of each y_i be the points $z_{i,j}$ where $1 \le i, j \le k$. In this case, the distance metric used is the Kolmogorov-Smirnov (KS) Statistic. The KS statistic is defined as

$$KS(m,n) = \sup_x |F_m(x) - F_n(x)| \tag{1}$$

where *sup* is the maximum over all values of x and F_m and F_n are the cumulative distribution functions (CDF) of the two samples m, n. So, the model computes:

$$\bar{x} = \frac{\sum_{i=1}^{k} KS(x, y_i)}{k} \tag{2}$$

$$\bar{y_i} = \frac{\sum_{j=1}^{k} KS(y_i, z_{i,j})}{k} \tag{3}$$

where $\bar{x}$ and $\bar{y_i}$ are the average distance from the new point to its k neighbors and the average distance from the new point's i-th neighbor to its k neighbors, respectively.

To determine if a new data point is unknown relative to the known application traffic, the model compares the average distance from the new point to its neighbors with the average distances from the new point's neighbors to their neighbors (Figure 2). However, to tune the model for optimal performance, a threshold, α, is used as a multiplier for the average distances between the new point's neighbors and their neighbors. This value will vary depending on the data set, and should be determined using cross-validation. So, the model determines the relations

$$\bar{x} > \alpha \bar{y_i} \tag{4}$$

for the k neighbors (y_i) of x. If this relation is true for each y_i, then the application session data point x is classified as unknown, otherwise it is classified as known.

For example, consider that for some given data set, an optimal value of α may be 2. This would mean that for a new data point to be classified as unknown, it is required for the average distance from the new point to its neighbors be twice as large as each of the average distances from these neighbors to their k nearest neighbors. If this is the case, then the traffic would be classified as unknown, and network management staff would be notified to review the finding. The k-Nearest Neighbors model can be updated over time with additional data to provide a more robust model. Operationally, incoming data which is classified as known by the model may be able to be incorporated into the model automatically, although this could have undesired effects on the model. An exploration of this is left to future work.

4 DESCRIPTION OF DATA

The data used for training and evaluating the machine learning models were originally captured as tcpdump output files in PCAP format. While, netflow data is not used here specifically, the data is still based upon flows of traffic. That is, the data is based upon flows and their statistics as opposed to deep packet inspection or other more invasive data collection methods. PCAP data files were then transformed into CSV files of packet dissections. The data were gathered using machines on the Emulab cluster, and tcpdump was run with controlled filters such that only desired traffic was captured, while SSH, Emulab network management and other undesired traffic was dropped. Data from four different application types were gathered, namely Skype, Google Hangouts (which has been used previously to study application classification [8]), YouTube and HTTP web browsing. A unique output file was generated for running individual sessions of each application.

4.1 Data Processing

In order to extract meaningful features for a machine learning model, each session file was processed into a collection of data points. Recall that a session file is a large set of individual packet dissections over a time period on the scale of minutes. Though each file represents one session of a running application, these files potentially contain several network connections, each defined by their own unique five-tuple (source and destination IP, source and destination port, and protocol). To represent the data in a way such that both time based and connection based features could be extracted, the data was split based upon both of these structures. In particular, a session file was split into many windows of 2 seconds each. Then for each 2 second window, individual data points were created for each five-tuple connection by aggregating all packet dissections into a single point, one point for each unique five-tuple connection (Figure 3). In this figure, the horizontal axis represents time in windows of 2 seconds, and the vertical axis represents distinct 5-tuples within a session, and the packets arriving or being sent as part of that connection. As is evident in the figure, some connections will have one or more 2-second windows with no packet arrivals. Rather than adding noise to the data, this actually helps represent variations in traffic flow levels for each distinct application and these attributes are captured in the features. Moreover, once these data points have been organized appropriately, all feature vectors could then be extracted for individual data points.

4.2 Feature Selection and Data Analysis

Features were developed through several iterations of experiments to find an optimal feature set. This feature set is useful for uncovering the differences in the data used in this study, which varies according to volume, consistency, and direction of traffic. However, features should be analyzed uniquely for individual operational systems to provide best performance. Operational systems may use all, some or none of these features, and may be augmented by other features, depending on the profiles of known traffic that would be used in the model. Using individual data points after processing, the following features were extracted to use in machine learning models:

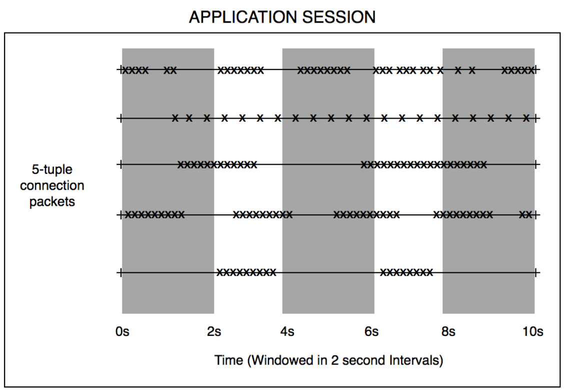
APPLICATION SESSION
5-tuple connection packets
0s
2s
4s
6s
8s
10s
Time (Windowed in 2 second Intervals)

Figure 3: Separation of a session network flow data into individual points through windowing

(1) Total number of packets sent
(2) Total number of bytes sent
(3) Average inter-arrival time of packets
(4) Average bit-rate of connection
(5) Largest packet size
(6) Smallest packet size
(7) Longest time between packet arrivals
(8) Shortest time between packet arrivals
(9) Direction of travel in Emulab (inbound or outbound)
(10) Number of ARP packets
(11) Number of DNS packets
(12) Number of TCP ACKs
(13) Minimum advertised receive window
(14) Maximum advertised receive window

These features are utilized in the classification methods and evaluation. Note that some features only apply to TCP type traffic. In order to handle traffic which is not TCP, these features are engineered such that values are representative of each protocol. For example, UDP traffic will always have 0 for the number of ACKs, and the minimum and maximum congestion window sizes will be zero and infinity, respectively. These features of data can be represented by a variety of CDFs and other charts that help to visualize the data before being used in machine learning models.

Data analysis with this feature set revealed keys for distinguishing between applications. As noted previously, each individual session of data was processed into a set data points, each of which was extracted into a feature vector. Moreover, an individual application session is represented as a list of many feature vectors. Graphing CDFs of all applications (with all sessions included) revealed that there are noticeable differences between applications. In some cases, each application is quite distinct from all other applications over the whole spectrum of values for an individual feature (Figure 4). In other cases, it is possible to distinguish each application from the others, but the CDF of values from different applications are intertwined across the spectrum of values (Figure 5). In other words, at some points along the CDFs, two applications may have similar curves, while at other points the applications diverge from each other.

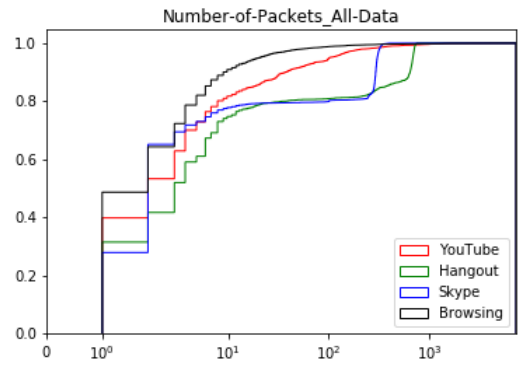

Figure 4: CDF of many averaged sessions from all application types

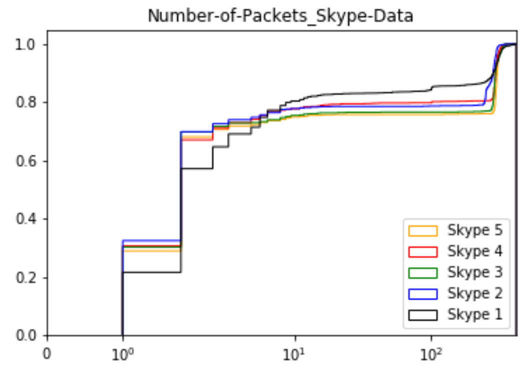

Figure 5: CDF of many individual sessions of a single application type

While these types of differences between applications are not easily distinguishable using some metrics, such as standard deviation and mean or Euclidean distance, it is possible to make the distinctions using a distance function that is appropriate for identifying the differences apparent in the CDFs. In this case, the most appropriate distance function is the two-sample KS statistic and is used in this work. Because each new application session is actually represented by several features, and each of these is represented by a conglomeration of sub-data points derived from 2-second intervals of individual 5-tuple connections, the KS statistic between two application sessions (shown as $KS(m,n)$ in the previous section), is computed by averaging the KS statistics for of each feature. That is, if there are l features (14 in this model), then more precisely,

$$KS(m,n) = \frac{\sum_{a=1}^{l} \sup_x |F_{m_a}(x) - F_{n_a}(x)|}{l} \quad (5)$$

where m_a is the a-th feature of the application session data point m.

5 METHODOLOGY

To implement the architecture as described previously, the data collection just described represents the collection of data at the edge of a network. In this case, the network in question is the Emulab cluster that houses the machines used for collection. Because the specific IP addresses of the machines used for data collection are known, the data accurately represents the architecture in which data is being sent in and out of the network. After the data was collected, it was appropriately transformed into featurized data that could be used in evaluation, precisely how it would happen in an operational network, except that this data processing didn't happen in real-time.

Recall that there are 4 unique application types represented in the data, namely Skype, Google Hangouts, YouTube, and general web browsing. In order to implement the architecture setting and then evaluate the model in a variety of scenarios, the data is divided many ways to create new trials. For each trial, the invocation of the model includes one or more applications in a group of data which is simulated as known data. This data is included in the final model for testing a holdout set for accuracy. The remaining application data is simulated as unknown data. A series of 150 experiments were run to evaluate the performance of the model. Among the 150 experiments, 15 different formats were used. First, all data was included as known data. Next, 4 formats specified one application type as unknown and the remainder as known data. Next, 6 formats specified half of the applications as known and the other half as unknown, using different combinations of the applications. Finally, 4 additional formats used one type of application as known and the remainder as unknown data.

For each trial, 20 percent of the data is held out for testing only. The remaining 80 percent of the data is used for training only. Both of these sets contain both known and unknown data. For each trial, leave-one-out cross-validation is completed. This produces a threshold hyper-parameter which tunes the model to optimal performance. That is, a range of intuitively possible thresholds is selected. A model is trained using all but one data point of the training data. Then the model is evaluated with this left out data point. This is performed for every point of the training data for each possible threshold value. The results for each possible threshold are averaged. Then, a final model is created based upon only the known data in the training set and the best threshold hyper-parameter determined during cross-validation. Finally, the test set is used to evaluate the final model and gain measurements of overall accuracy, as well as specific performance with known and unknown traffic in the test set.

In order to compare the methods presented to a baseline, a strawman machine learning model is created by using the same architecture, but using a typical Euclidean distance instead of using a KS-statistic in the distance function for the k-Nearest-Neighbors model. This represents a method which is generic and can be used to model any data set, but doesn't necessarily represent the nuances of that data set accurately, particularly for high-dimensional data and data represented by features that are measured on different scales.

	Predicted Known	Predicted Unknown
True Known	94.72 %	5.28 %
True Unknown	81.07 %	18.93 %

Table 1: Confusion matrix for strawman (Euclidean) model

	Predicted Known	Predicted Unknown
True Known	91.98 %	8.02 %
True Unknown	7.19 %	92.81 %

Table 2: Confusion matrix for unknown traffic (KS-Statistic) model

6 RESULTS

6.1 Strawman Method Results

The strawman method described in the previous section achieves an accuracy of only 57.72 % on the entire data set, with 94.72 % accuracy on the known traffic but only 18.93 % accuracy on unknown traffic (Table 1)]. The most notable aspect of these results is that the strawman approach is obtains a dismal 18.93 % accuracy on the unknown data. This is a poor result given that a random classifier should be able to guess correctly up to 50 % of unknown data.

6.2 KS Statistic Method Results

Using the KS statistic modeling approach architected in this study instead of a typical Euclidean distance with the k-Nearest-Neighbors model yielded significantly better results. The improved method achieves an overall accuracy of 92.67 %. Furthermore, the KS statistic method achieved an accuracy of 91.98 % on the known traffic and 92.81 % on unknown traffic (Table 2). This method clearly outperforms the strawman method described above. One persistent weakness of this classification is the high false negative rate for operational endeavors. While overall improvement is still desireable, it is especially important to further improve the ability of the classifier to correctly identify unknown traffic. But, this method shows promise for high accuracy machine learning classification of unknown application traffic while mainting high accuracy of known traffic, and further improvement with more sophisticated and larger data sets is left to future work.

7 OPERATIONAL IMPLEMENTATION AND FUTURE WORK

The first concern to address before completing an operational implementation of this approach is to reduce the relatively high false negative rate shown in the results of this study. In order to reduce the false negative rate, it is necessary to gather significantly more data from the applications being considered. Then, for each application session that is incorrectly classified as a known application, it is necessary to examine the features that are having the greatest impact on the misclassification of that session. By gathering this information, changes and additions can be made to the feature set to create a better distinction between known and unknown traffic. Creating a cleaner separation between the known and unknown traffic with a more refined feature set will reduce this false negative rate.

There are a number of other operational considerations to be made for this type of machine learning approach to network security. Assuming that a controlled, initital data set is already available, network management staff can move ahead with creating a model for evaluating incoming traffic. However, incoming traffic is diverse compared with the data used to train and evaluate the k-Nearest Neighbors approach. This approach must be evaluated in further phases to determine the accuracy of the algorithm with uncontrolled data. But, in order to complete this step, it is requisite to have a method to separate the large amount of incoming traffic into individual sessions of distinct applications, as this is the format with which the machine learning model has been trained. An intuitive approach will likely separate traffic based upon five-tuple connections and then combine multiple connections into a session, perhaps based upon the IP address of the client within the network being monitored. This is a critical step that must still be explored to make this approach operationally viable.

Another critical aspect of this approach is the evolution of the model to adapt to changing natures of approved applications. In an ideal operational setting, the data used to create the model would be augmented in an automated way based upon other traffic approved by the model. However, this could cause the model to shift in ways such that truly unknown traffic begins to fit within the known traffic of the model. The model could also change in other detrimental ways, and automated methods for updating the model must be thoroughly explored and tested to find a way to update the model to include changes in known traffic.

Finally, to enhance the utility of the approach, it is possible to combine this approach that can classify unknown application traffic with methods that are proven to accurately classify known bad application traffic (e.g., specific types of well-known and profiled malware). This could be achieved by placing two separate machine learning models in one data pipeline. First, data could be compared with a model to identify any known inappropriate application traffic. If this test passes, then traffic can be sent to the model developed in this study which can identify unknown traffic.

8 CONCLUSION

This study has explored and proven the utility of a machine learning approach to be able to classify unknown traffic in a computer network. The approach achieved up to 92.67 % accuracy on a controlled data set with a basic feature set. The study also discusses operational factors related to the machine learning approach, and how they might initially be studied to find optimal solutions for an operational setting. While the machine learning approach presented is limited by only using controlled data and using a basic feature set, the approach shows potential for effectively solving the problem of securely handling unknown applications in a network. This study shows that the machine learning approach presented warrants a full operational research study to realize the full potential of the approach.

ACKNOWLEDGMENTS

The authors would like to thank the University of Utah Office of Undergraduate Research for funding and support for this project. The authors would also like to thank the Flux Research Group and Emulab staff for the ability to gather data for this research. This material is based upon work supported by the National Science Foundation under Grant No. 1642158.

REFERENCES

[1] 2016. IBM QRadar. https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WGW03211USEN. (2016).

[2] 2016. Juniper Machine Learning. https://go.juniper.net/assets/pdfs/OSI/Outsmarting_Malware.pdf. (2016).

[3] 2017. LogRhythm Machine Learning. https://logrhythm.com/employing-machine-learning-in-a-security-environment-white-paper/. (2017).

[4] 2017. scikit-learn. http://scikit-learn.org/. (2017).

[5] Tony Antonio and Adi Suryaputra Paramita. 2014. Feature selection technique impact for internet traffic classification using naïve bayesian. *Jurnal Teknologi* 20 (2014), 85–88.

[6] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. 2006. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*. ACM, 6.

[7] Selim Ciraci, Boon Thau Loo, Assaf Schuster, Geoff Outhred, et al. 2016. Taking the blame game out of data centers operations with netpoirot. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 440–453.

[8] Jayeeta Datta, Neha Kataria, and Neminath Hubballi. 2015. Network traffic classification in encrypted environment: a case study of google hangout. In *Communications (NCC), 2015 Twenty First National Conference on*. IEEE, 1–6.

[9] Walter De Donato, Antonio Pescapé, and Alberto Dainotti. 2014. Traffic identification engine: an open platform for traffic classification. *IEEE Network* 28, 2 (2014), 56–64.

[10] Yu-ning Dong, Jia-jie Zhao, and Jiong Jin. 2017. Novel feature selection and classification of Internet video traffic based on a hierarchical scheme. *Computer Networks* 119 (2017), 102–111.

[11] Pietro Ducange, Giuseppe Mannarà, Francesco Marcelloni, Riccardo Pecori, and Massimo Vecchio. 2017. A novel approach for internet traffic classification based on multi-objective evolutionary fuzzy classifiers. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 1–6.

[12] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. 2006. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*. ACM, 281–286.

[13] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson. 2007. Semi-supervised network traffic classification. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 35. ACM, 369–370.

[14] Jerome Francois, Shaonan Wang, Walter Bronzi, Radu State, and Thomas Engel. 2011. Botcloud: Detecting botnets using mapreduce. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*. IEEE, 1–6.

[15] Nen-Fu Huang, Che-Chuan Li, Chi-Hsuan Li, Chia-Chi Chen, Ching-Hsuan Chen, and I-Hsien Hsu. 2017. Application identification system for SDN QoS based on machine learning and DNS responses. In *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*. IEEE, 407–410.

[16] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: multilevel traffic classification in the dark. In *ACM SIGCOMM Computer Communication Review*, Vol. 35. ACM, 229–240.

[17] Hyunchul Kim, Kimberly C Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. 2008. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT conference*. ACM, 11.

[18] Jeankyung Kim, Jinsoo Hwang, and Kichang Kim. 2016. High-performance internet traffic classification using a Markov model and Kullback-Leibler divergence. *Mobile Information Systems* 2016 (2016).

[19] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. 2013. A survey of network flow applications. *Journal of Network and Computer Applications* 36, 2 (2013), 567–581.

[20] Wei Lu and Ling Xue. 2014. A heuristic-based co-clustering algorithm for the internet traffic classification. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*. IEEE, 49–54.

[21] Andrew W Moore and Denis Zuev. 2005. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33. ACM, 50–60.

[22] Thuy TT Nguyen and Grenville Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials* 10, 4 (2008), 56–76.

[23] Yafei Sang, Shicong Li, Yongzheng Zhang, and Tao Xu. 2016. Prodigger: Towards robust automatic network protocol fingerprint learning via byte embedding. In *Trustcom/BigDataSE/IâĂŇ SPA, 2016 IEEE*. IEEE, 284–291.

[24] Yafei Sang, Yongzheng Zhang, Yipeng Wang, Yu Zhou, and Xu Tao. 2014. A segmentation pattern based approach to automated protocol identification. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2014 15th International Conference on*. IEEE, 7–12.

[25] Davide Sanvito, Daniele Moro, and Antonio Capone. 2017. Towards traffic classification offloading to stateful SDN data planes. In *Network Softwarization (NetSoft), 2017 IEEE Conference on*. IEEE, 1–4.

[26] Brian Schmidt, Ala Al-Fuqaha, Ajay Gupta, and Dionysios Kountanis. 2017. Optimizing an artificial immune system algorithm in support of flow-Based internet traffic classification. *Applied Soft Computing* 54 (2017), 1–22.

[27] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. 2004. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*. ACM, 512–521.

[28] Wazen M Shbair, Thibault Cholez, Jerome Francois, and Isabelle Chrisment. 2016. A multi-level framework to identify https services. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 240–248.

[29] Alok Tongaonkar, Ram Keralapura, and Antonio Nucci. 2013. Santaclass: A self adaptive network traffic classification system. In *IFIP Networking Conference, 2013*. IEEE, 1–9.

[30] Alok Tongaonkar, Ruben Torres, Marios Iliofotou, Ram Keralapura, and Antonio Nucci. 2015. Towards self adaptive network traffic classification. *Computer Communications* 56 (2015), 35–46.

[31] Yu Wang, Yang Xiang, Jun Zhang, Wanlei Zhou, Guiyi Wei, and Laurence T Yang. 2014. Internet traffic classification using constrained clustering. *IEEE transactions on parallel and distributed systems* 25, 11 (2014), 2932–2943.

[32] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. 2008. Internet traffic behavior profiling for network security monitoring. *IEEE/ACM Transactions On Networking* 16, 6 (2008), 1241–1252.

[33] Xinyan Yang, Yunhui Yi, Xinguang Xiao, and Yanhong Meng. 2018. Mobile Application Identification based on Hidden Markov Model. In *ITM Web of Conferences*, Vol. 17. EDP Sciences, 02002.

[34] Sebastian Zander, Thuy Nguyen, and Grenville Armitage. 2005. Automated traffic classification and application identification using machine learning. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*. IEEE, 250–257.

[35] Hongli Zhang, Gang Lu, Mahmoud T Qassrawi, Yu Zhang, and Xiangzhan Yu. 2012. Feature selection for optimizing traffic classification. *Computer Communications* 35, 12 (2012), 1457–1471.

[36] Jun Zhang, Yang Xiang, Yu Wang, Wanlei Zhou, Yong Xiang, and Yong Guan. 2013. Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed Systems* 24, 1 (2013), 104–117.

[37] Shuyuan Zhao, Yongzheng Zhang, and Peng Chang. 2017. Network Traffic Classification Using Tri-training Based on Statistical Flow Characteristics. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE, 323–330.

[38] MIN-JIE ZHU and NAI-WANG GUO. 2017. Abnormal Network Traffic Detection Based on Semi-Supervised Machine Learning. *DEStech Transactions on Engineering and Technology Research* ecame (2017).