Application of Deep Learning on the Characterization of Tor Traffic using Time based Features

Clayton Johnson[1], Bishal Khadka[1], Ethan Ruiz[1],
James Halladay[1], Tenzin Doleck[2], and Ram Basnet[1]*
[1]Colorado Mesa University, Grand Junction, Colorado, USA
{cpjohnson, bkhadka2, elruiz, jehalladay}@mavs.coloradomesa.edu, rbasnet@coloradomesa.edu
[2]Simon Fraser University, Burnaby, CA
tdoleck@sfu.ca

Received: August 17, 2020; Accepted: February 11, 2021; Published: February 28, 2021

Abstract

The Onion Router (Tor) is a popular network, widely used by both political dissidents and cyber criminals alike. Tor attempts to circumvent government censorship and surveillance of individuals by keeping secret a message's sender/receiver and content. This work compares the performance of various traditional machine learning algorithms (e.g. Random Forest, Decision Tree, k-Nearest Neighbor) and Deep Neural Networks on the ISCXTor2016 time-based dataset in detecting Tor traffic. The research examines two scenarios: the goal of Scenario A is to detect Tor traffic while Scenario B's goal is to determine the type of Tor traffic as one of eight categories. The algorithms trained on Scenario A demonstrate high performance, with classification accuracies $> 99\%$ in most cases. In contrast, Scenario B yielded a wider range of classification accuracies (40-82%); Random Forest and Decision Tree algorithms demonstrate performance superior to k-Nearest Neighbors and Deep Neural Networks.

Keywords: Tor traffic, deep learning, machine learning, traffic identification, encrypted traffic

1 Introduction

The internet has revolutionized the world and no industry has been untouched by the changes brought forth by the technology. Two key technologies the internet is built upon are Internet Protocol (IP) and Transmission Control Protocol (TCP) together which allow for reliable high-speed communication between connected devices. IP is a network layer protocol used to deliver packets to client devices and uses a unique address, known as an IP Address, to indicate the destination of incoming packets. TCP sends packets to devices in a predictable and reliable manner through establishing connections that verify whether sent packets arrive at their destination, resending the packets if they do not arrive. As the Internet has provided many benefits to the mass around the world, the same has also been constantly exploited by cybercriminals [22][23][13].

In the emerging era of the Internet, privacy has become a chief concern among its users. Many professionals who are critical to governance such as journalists, activists, and detectives can potentially have their work tampered with, their investigations suppressed, and operations interrupted if their privacy is compromised. The Onion Router (Tor) was released in 2004 in order to address the many privacy concerns that non-technical day-to-day internet users previously had little control over [34]. According

*Corresponding author: Department of Computer Science and Engineering, Colorado Mesa University, 1100 North Aveneue, Grand Junction, CO 81501 USA. Tel: +1 970 248 1682 Web: `https://rambasnet.github.io`

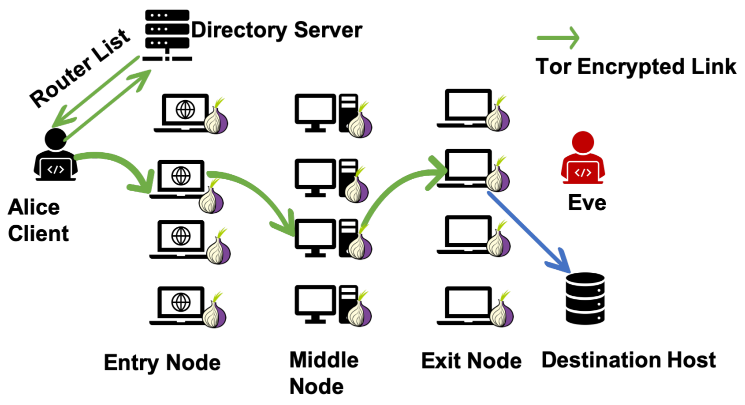

Figure 1: Tor Network and Traffic Flow

to the Tor Project [10], Tor implements the onion routing scheme providing maximum privacy to the communicating parties over the Internet. The encrypted traffic is routed via many servers (analogous to onion layers) maintaining the perfect forward secrecy and privacy [34][10].

Tor is a circuit-based low-latency service that can anonymize traffic over applications that use TCP to communicate [34]. Tor lets users access websites without revealing their logical or physical locations to those sites or outside observers [10]. Tor has various relays where sensitive traffic enters from the entry node and the traffic gets encrypted in the internal nodes or diverse relays in numerous Tor servers and gets out of the exit node. In this way, it is burdensome for third-parties to know where the traffic was coming from and what it is requesting for thus maintaining anonymity as well as privacy.

With the advent of Tor Browser, Tor has become more accessible to regular internet users. Tor Browser has become instrumental in providing Internet users with the highest level of anonymity and unbreakable encryption in order to access critical resources, social media, and websites that may be typically censored by nation states as witnessed during the Arab Spring in late 2010 [34][10].

Tor Browser works by having a 3-layer proxy. This allows for safe and anonymous data traffic transmission. Tor browser connects randomly to a publicly listed entry node from the directory server, bounces that traffic through a randomly selected middle relay, then it finally outputs the traffic through the final exit node. Once the connection to the destination host is established, the secure and private circuit is formed and the communication traffic flows back and forth using the same circuit during the established session. Exit node doesn't use the Tor encryption for the final segment of the connection to the destination host, leaving it up to the host or the service provider. This is usually referred to as Onion Routing and is the main functionality of Tor Browser. Onion routing is implemented by encryption in each layer as well as at the application layer which is part of the communication (TCP/IP) protocol stack [32]. Each node only knows the previous and the next node providing a local view of the complete connection. Node is also referred to as the router. Figure 1 depicts a typical Tor network and traffic relay.

For our research, we used various traditional machine learning algorithms and deep-learning techniques on time-based features to identify Tor traffic. More specifically, we wanted to classify Tor traffic from regular internet traffic. The focus of our experiments has been split into two scenarios. The experiments on scenario A demonstrate that time-based features can be effective identifiers for a variety of machine-learning algorithms. The results using deep-learning techniques match or exceed the accuracy of other techniques. Scenario B's results demonstrate how Tor traffic can still be vulnerable to certain introspection techniques. Despite the traffic being encrypted, the experiments demonstrate that it is possible to classify the content of the traffic based on inherent differences between the types of contents being delivered.

This paper reveals the shortcomings of the Tor protocol when it comes to hiding Tor traffic. We also compare the performance of deep-learning techniques to other traditional machine-learning techniques that have previously been applied to this problem. Differentiating Tor from non-Tor traffic is shown to be a solved issue. Further advances toward the classification of the content of Tor traffic are required.

2 Related Work

Lashkari et al. [26] presented the ISCXTor2016 dataset and reported the performance of multiple machine learning algorithms on the binary classification task of Tor versus non-Tor packet data and multi-classification task determining the type of data (audio-streaming, browsing, chat, file transfer, mail, peer-to-peer, video-streaming, voice-over-ip) within Tor packets. The work reported that ZeroR, C4.5 decision tree, and k-Nearest Neighbors (kNN) models performs as high as 0.99 precision and recall for the binary classification problem of classifying Tor from non-Tor traffic. Their multi-class classification using Random Forest (RF), Decision tree (C4.5), and k-Nearest Neighbor (kNN) learning algorithms yields a wider range of performance metrics from 0.60 – 0.84 precision and recall. Their primary contribution is that they generated and labelled the dataset dubbed ISCXTor2016 making it publicly available.

Rao et al. [36] presented an improved clustering algorithm called the Gravitational Clustering Algorithm (GCA) to identify Tor packets with high accuracy. The dataset contains packet data collected from ExperimenTor, a testbed for application testing on an isolated Tor network [14]. This work reported an average accuracy from GCA on the dataset of 80%, outperforming K-means, Expectation-Maximization (EM), and Density-based Spatial Cluster of Applications with Noise (BDSCAN). Additionally, with the introduction of a new feature, packet length distribution, GCA increases its average accuracy by almost 10%. The main contributions of [36] are the proposal of GCA and this new feature.

Soleimani et al. [38] also reported very high-performing models that accurately identify Tor packets within the first 10-50 packets using only a few statistical features. Using Adaboost, RF, C4.5, and a Support Vector Machine (SVM) on a dataset containing Tor traffic from Obfs3, Obfs4, and ScrambleSuit Tor pluggable transports and background traffic, these models perform nearly perfect due to the predictable nature of the set-up sequence for Tor packets. The features that yielded highest performance are total flow volume, mean packet length, and standard deviation of packet length.

Aminuddin et al. [12] presented a survey of the applications of machine learning in the identification and classification of Tor packets from 2012 to 2018. They report that most of the methods used in the field utilize flow and packet features, while circuit properties are used by relatively few research. Almost all previous research used supervised models, whereas only a few used semi-supervised or unsupervised models. The most-common methods in the field include C4.5, SVM, Naive Bayes, Bayesian Networks, and RF. Few of the surveyed works (35%) used real-time methods or are tested in real-time. Fewer than 40% of the datasets used in the surveyed works are public and just over half of the methods used are compatible with Tor [12].

Zhen Ling et al. [27] integrated a Tor exit router with an Intrusion Detection System (IDS) to detect and classify malicious traffic routed through the exit router. The implementation, TorWard, then redirects outgoing Tor traffic back into the Tor network in order to shield users from legal trouble. It was noted that this could have detrimental effects on the performance of the Tor network. Analyzing the data provided by TorWard, the team finds that up to 10% of Tor traffic triggered IDS alerts, discovering more than 200 different types of malware. Half of all Tor traffic was identified as Transport Layer Security (TLS) traffic, which may be indicative of the growing popularity of Tor plugins and browsers.

Abdelberi Chaabane et al. [20] provided an in-depth analysis of the Tor Network through several exit nodes using Deep Packet Inspection (DPI) technique to classify the traffic exchanged through the nodes. The team employed cautionary privacy measures to restrict data analysis to happen on the fly with no storage for exit logging, whereas, for entrance logging, the IP address connecting to the Tor node is recorded. The analysis found that a large portion of Tor traffic is Peer-to-Peer (P2P) content and went further to separate the traffic into categories based on which protocol is being used to deliver the content over Tor.

Youting Liu et al. [28] used Intermittent Traffic Pattern (ITP) related features to detect video traffic over encrypted web traffic. The work proposed using ITP features along with the kNN to differentiate video from non-video traffic. A variety of unstandardized protocols, employed to deliver video flows, posed a challenge to the team's experiments. Due to traffic encryption, traditional methods such as Deep Packet Inspection (DPI) are not available to identify video-based traffic. However, intermittent traffic patterns are caused by fragmented transmissions, which are common features of many different video streaming protocols. ITP features allow systems to identify video traffic regardless of whether the traffic has been encrypted.

Kota Abe et al. [11] proposed a new method for launching a fingerprinting attack to analyze Tor anonymity using a stacked denoising autoencoder (SDAE). They used the uWaterloo dataset that contains 100 monitored websites and 9,000 unmonitored sites. The method for testing was done by first having an attacker collect training data for machine learning. Then the attacker accesses websites he or she wants to monitor through Tor fingerprinting and then captures the traffic data multiple times. The attacker also collects traffic data from many other websites. The attacker then extracts the Tor cells from the data and it is then used as an input for the autoencoder. They then sort out the data to feed as an input into the input vector for model training. The input vector took a simple input as 1, -1, or 0 and the results show 88% accuracy.

Park et al. [31] covered the different protocols that are used in peer-to-peer application traffic when using Tor browser. The team identified different types of protocols used by applications and Tor traffic usage. This paper used what is called a Traffic Clustering Scheme which refers to traffic workload characteristics rather than protocol decomposition. They also use an Application Breakdown Scheme which tries to identify the application being used when sending traffic. The traffic classification can classify various types of traffic that are generated by a single application. The LCS-based (Longest Common Subsequence) Application Signature Extraction (LASER) [32] algorithm requires sanitized packet collection as its input data. The sanitized packets refer to packets belonging to the target application only. The collecting agent divides the sanitized packets depending on each flow and stores them in a separate packer dump file. The method for testing was done using the proposed classification scheme. They chose an application called Fileguri which provides web browsing, searching, downloading, etc. The classification accuracy results on the downloading portion were superior $(70\% - 90\%)$ compared to web browsing $(12\% - 14\%)$.

Overall, the literature clearly indicates there has been substantial effort to detect and classify Tor traffic across many datasets. As discussed in [12], there is a lack of a standardization for Tor traffic datasets and a wide array of machine learning algorithms implemented with relatively high performance. Despite this, there is little to no published work on the applications of deep learning within the context of detecting and classifying Tor network traffic. Additionally, few works attempt to identify the characteristics of Tor traffic that allow for such high detection rates. This work serves to investigate the applications of deep learning and the presented features within the context of detecting and classifying Tor network traffic.

3 Our Approach

This section contains multiple subsections with the goal of thoroughly exploring all the steps taken to gather data, train models, and evaluate the performance of various traditional machine-learning and deep-

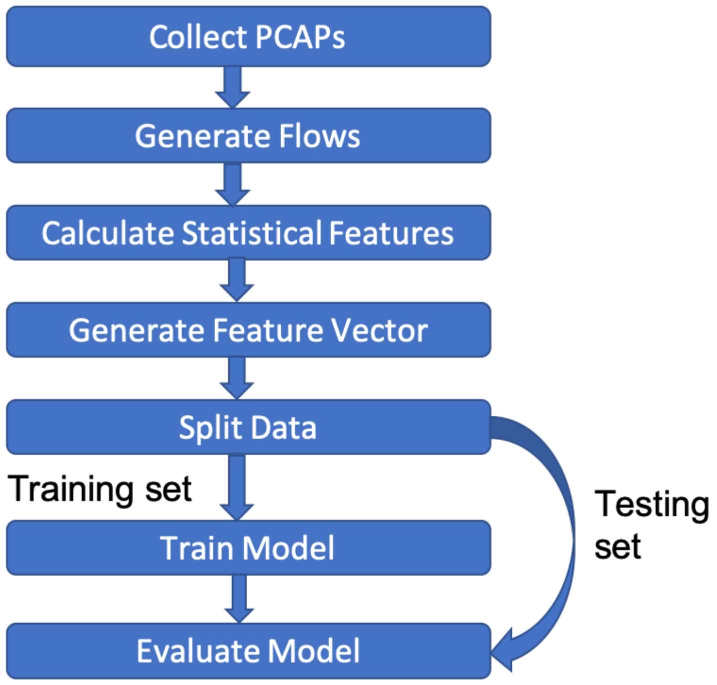

Figure 2: Experimental Workflow

learning models. Experimental workflow is presented first, followed by sections examining the dataset, discussing the Python modules, and the metrics used to present the results.

3.1 Experimental Workflow

The ISCXTor2016 [26] dataset contains time-based traffic features from real-world Tor traffic collected through the Whonix Linux operating system. Whonix is based on Kicksecure, another security-driven OS, that provides an isolated workstation and gateway for Tor usage by default [7]. Figure 2 depicts our experimental workflow for binary and multi-class classification experiments to train and test machine learning models.

The outgoing Tor network traffic from the Whonix workstation is sniffed using both Wireshark and tcpdump. This packet sniffing process generates PCAP (packet capture) files that can be analyzed using a variety of methods. In this process, Lashkari et al. [26] analyzed the PCAP files using the ISCXFlowMeter (now CICFlowMeter) program from the University of New Brunswick.

This network traffic analyzer program generates traffic flows and calculates more than 80 statistical features using the provided PCAPs [5]. A flow is defined as a series of packets that contain the same source/destination IPs, source/destination ports, and protocol. Due to conflicts with Tor's crypto relay protocols and security design (i.e. preventing end-to-end tagging attacks and congestion issues), Tor does not support UDP and will not for the foreseeable future [8]. As such, all the analyzed flows are TCP. Once the flows are generated and these additional features are calculated, the data is separated into training and testing datasets.

Stratified 10-fold cross validation is used to train the models on the given dataset. The stratification of the folds ensures the proportionality of the target classifications in each fold is representative of the overall dataset. This method was chosen over a validation dataset due to the low frequency of some target classifications in Scenario B, allowing the team to better use all the data available for training and testing.

Table 1: Data Composition of Scenario A

File	Tor	NonTor	Total
10s	8,044	59,790	67,834
15s	3,314	18,758	22,072
30s	1,771	14,651	16,422
60s	914	15,515	16,429
120s	470	10,782	11,252

Generally, the usage of 10 stratified folds appears to minimize the variance-bias problem compared to other k-values or leave-one-out methods [18][25]. While every model is only trained on nine folds and tested with the remaining tenth, the reported accuracy of each learning algorithm is the average performance across all ten testing folds. Additionally, all confusion matrices and receiver operating characteristic (ROC) curves are derived from the final fold models. This is because the learning model objects were explicitly recreated and initialized for every fold to ensure the models were not training on all the training data.

3.2 Dataset

A proper data set is pivotal in the field of data science. Traditional data (categorical, numerical) or big data (categorical, numerical, digital audio/video signals, etc.) are labeled, scrubbed or cleansed, dealt with missing values, and balanced in order to obtain the data set in which a precise prediction can be made. Using those obtained data sets, data scientists will be able to obtain the highest predictive accuracy.

In this experiment, we use ISCXTor2016 [26] dataset. The dataset is plentiful and clean with very few missing values or non-numeric values such as infinity and nan. The ISCXTor2016 dataset comprises two scenarios: A and B that helped us experiment with binary and multi-class classification of Tor vs non-Tor traffic as well as various categories of applications. The ISCXTor2016 dataset is publicly available from the University of New Brunswick-Canadian Institute for Cybersecurity website [5].

In Scenario A, each sample is labelled as either Tor or nonTor traffic. In total, Scenario A contains five datasets, each extracted using variable timeout values (10s, 15s, 30s, 60s, and 120s). The composition of each dataset is provided in Table 1. Scenario A dataset is used as a binary-class classification problem. Table 1 provides the number of samples that are presented in various files where each file represents the data flow for various variable timeouts in seconds.

As seen in Table 1, Scenario A is imbalanced with the minority class composing $< 5\%$ of the data in the extreme case. Because of this imbalance and as seen in the results, even a ZeroR solution would achieve 95% accuracy and it's likely that any model slightly more complex than ZeroR would perform even higher. With the goal of presenting performance metrics with respect to a low-performing baseline model, we created a separate dataset by downsampling Scenario A to balance the Tor and nonTor classes. However, consequently, the balanced dataset is much smaller than the original dataset, allowing for bias to be introduced. Table 2 below presents the dataset composition of the balanced Scenario A dataset.

All the data traffic presented (see Table 3) in Scenario B is Tor, however the labels indicate the actual type of traffic for each sample. The types of Tor traffic are split into eight categories: Audio-Streaming, Browsing, Chat, E-mail, File-Transfer (FTP), Peer-to-Peer (P2P), Video-Streaming, and Voice-over-IP (VoIP). Similar to Scenario A, Scenario B contains five datasets, each extracted using variable timeout values (10s, 15s, 30s, 60s, and 120s). Overall, the ISCXTor2016 dataset contains 10 original data files, with five additional files being created for Balanced Scenario A.

Table 2: Data Composition for Balanced Scenario A

File	Tor	NonTor	Total
10s	8,044	8,044	16,088
15s	3,314	3,314	6,628
30s	1,771	1,771	3,542
60s	914	914	1,828
120s	470	470	940

Table 3: Data Composition for Scenario B dataset

File	FTP	Browsing	Video	Audio	VoIP	Chat	P2P	Mail	Total
10s	864	1,604	874	721	2,291	323	1,085	282	8,044
15s	480	227	598	46	1,509	243	71	186	3,360
30s	246	133	345	32	758	147	38	104	1,803
60s	125	73	177	22	381	84	20	54	936
120s	63	41	90	16	193	45	10	28	486

While the ISCXTor2016 dataset is originally in the attribute-relation file format (ARFF), it was converted to comma-separated values (CSV) for ease of use outside of the Weka [6] framework. This conversion does not change the data. There are 23 features derived and used from the CICFlowMeter program such as forward and backward inter-packet arrival times, flow arrival times, time active and idle, bytes and packets per second, with statistical features (std, min, max, and mean), and the duration of a flow. While CICFlowMeter can provide more than 80 features from PCAP data files, this work is focused on time-based features. Additionally, we are only using the features selected by Lashkari et al. [26] to decrease noise and the feature-set in the dataset.

3.3 Features

Each of the cleaned dataset contains 79 features; out of which 2 (*Destination Port and Protocol*) are treated as categorical using 1-to-n encoding and the rest are all numeric. The original dataset also contains data with best feature selection; we do not use this data in our experiments, however. Some examples of the numeric features are *Timestamp, Flow Duration, SYN Flag Count, Packet Length Min, Packet Length Max*, etc. More details about all the features can be found in [36].

4 Machine Learning Frameworks

All the modules described below were chosen for their accessibility, maintenance, and popularity within the machine learning and deep learning field.

4.1 Scikit-Learn

Scikit-Learn (sklearn) is an open-source module created to allow for usage of high-level machine learning algorithms in the Python programming language [33]. Due to its consistent API, combined with the high-level nature of Python, scikit-learn is extremely accessible for “non-experts” in machine learning.

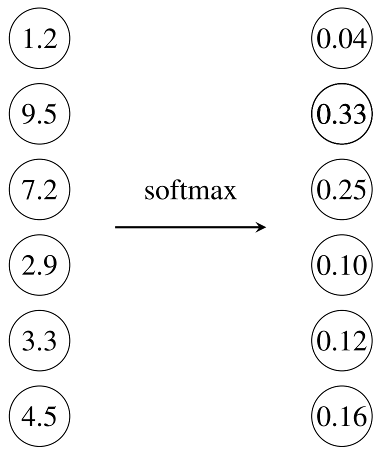

Figure 3: Softmax Function Example

The sklearn classifiers used in this work include the DummyClassifier, RandomForestClassifier, DecisionTreeClassifier, and kNeighborsClassifier. In the original work, Lashkari et al. [26] use Weka [6] for the classification of the scenarios. The Weka-based classifiers used are ZeroR, C4.5, and k-Nearest Neighbors algorithms for Scenario A and Random Forest [17], C4.5, and k-Nearest Neighbors algorithms for Scenario B. Scikit-Learn's Random Forest classifier implements a soft-voting mechanism instead of a hard-voting mechanism as described in [4][3]. Additionally, instead of implementing a forest of ID3 trees, sklearn uses a forest of Decision Trees (DT), which implement an optimized version of the CART algorithm as opposed to C4.5 [3]. Additionally, the ZeroR classifier in Weka is implemented through the DummyClassifier in sklearn.

4.2 Keras

Keras is a deep learning framework for Python that allows for the implementation of both TensorFlow and Theano in the backend [2]. While the Keras framework is not as accessible as FastAI, it provides a high-level API with the goal of streamlining the process from an idea's conception to implementation. A Sequential model was used to create a deep neural network with multiple Dense layers in between. The team finds that the overall dimensions of the neural networks, while affecting the performance of the model, had a smaller effect than changing the activation functions and optimizers used. For the hidden layers within the model, the rectified linear unit (ReLU) activation function (equation 2) is used while the output layer utilizes the softmax activation function (equation 1). Softmax is typically used for multi-classification problems because the range for the output is between 0 and 1, serving to act as the likelihood for a given classification [29]. For gradient-based optimization, the model uses Adam due to its lower computational cost for training, and tolerance for noisy gradients [24].

$$f(\vec{x}_i) = \frac{e^{\vec{x}_i}}{\sum_{j=0}^{n} e^{\vec{x}_j}} \tag{1}$$

$$f(\vec{x}) = max(0,\vec{x}) \tag{2}$$

Figure 3 is an example demonstrating the softmax function implemented in Keras, where the input vector is on the left-hand side, containing six values. These input values may originate from data or a layer within the network preceding this layer. The output of the softmax function is on the right-hand side, showing that the values have been scaled down to the range $[0,1]$. These values may now be interpreted as the likelihood of each classification, as valued by the model. The darker cell is the chosen classification because it has the highest output value from the softmax function.

While most of the metrics described below are implemented in Keras, generating confusion matrices and receiver operating characteristic (ROC) curve and area-under-curve (AUC) metrics required additional effort to implement.

4.3 fast.ai

fast.ai is a Python module that implements a high-level API to a PyTorch backend. The goal of fast.ai is to easily allow for experts in other fields of science, such as virologists or astronomers, to implement popular deep learning techniques within their respective settings. This framework has seen multiple popular successes in research and industry [1].

Since the dataset is converted to CSV format, the fast.ai TabularList object is used to import the data and train a tabular_learner model. Due to the extremely high-level nature of fast.ai, there are few hyperparameters to adjust the model.

5 Performance Metrics

The various metrics used to evaluate the models are described below. Typically, the performance of the models is measured by accuracy (Acc), which is defined. This definition works for both binary and multi-classification problems.

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} = \frac{correct\ predictions}{total\ predictions} \tag{3}$$

, where TP, TN, FP, FN are the true positives, true negatives, false positives, and false negatives, respectively. While accuracy is helpful when trying to generally understand how a model is performing, this metric does not offer any other insights into the performance of a given model. To improve the general understanding of the model, sensitivity and specificity metrics are used.

Sensitivity (or Recall), ranging from $0-1$, is inversely proportional to the number of false negatives predicted by the model. On the other hand, specificity, ranging from $0-1$, is inversely proportional to false positives. Precision is another metric commonly used that, similar to specificity, decreases with increases in false positives.

$$Sensitivity = \frac{TP}{TP+FN} \tag{4}$$

$$Specificity = \frac{TN}{TN+FP} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

The F1-score metric uses precision and recall (sensitivity) to evaluate a model. The equation for F1-score is below.

$$F1 = 2 * \frac{precision \times recall}{precision + recall} \tag{7}$$

The trade-off between sensitivity and specificity indicates some sort of optimal model performance. The receiver operating characteristic (ROC) curve shows the trade-off between sensitivity and specificity by presenting a curve with the cumulative distribution of sensitivity on the y-axis and cumulative distribution of 1-specificity on the x-axis. From the ROC curve, another metric arises: area under the curve (AUC), which ranges from $0-1$ and represents the area under the ROC curve. Generally, the model with a ROC curve closest to the top-left corner, or with the maximum AUC, is the most-preferable. Model selection, however, depends on the problem. These concepts are more clearly illustrated when reporting the results of Scenario B in the Results section.

Two additional metrics, micro-average and macro-average, are variations on F1-scores displayed on ROC curves. The micro-average attempts to account for bias brought forth by imbalanced data while the macro-average calculates the F1-score without any regard to the frequency of a given classification [30]. The equations for these are shown in 8 and 9.

$$Macro\_average = 2\frac{(\frac{1}{n}\sum_{i=1}^{n}P_i)(\frac{1}{n}\sum_{i=1}^{n}R_i)}{(frac1n\sum_{i=1}^{n}P_i+\frac{1}{n}\sum_{i=1}^{n}R_i} \tag{8}$$

$$Micro\_average = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i+R_i}, \tag{9}$$

, where n is the number of classes, P_i is the precision of a given class, and R_i is the recall of a given class. In this work, we show both the micro- and macro-average ROC curves. The final technique used to report the performance of the deep learning models is the confusion matrix, which shows the performance of a model on test data by showing the predicted classification for a given sample next to the correct classification [9]. In conjunction with recall and precision, a confusion matrix easily demonstrates the shortcomings or strengths of a proposed model and workflow, as described in the Results section.

6 Results

In this section we present the results for both Scenario A and Scenario B experiments using our machine learning classifiers. For Scenario A, the report shows the accuracies for the unbalanced and balanced datasets. For Scenario B, accuracy, ROC, AUC, and confusion matrix metrics are presented to thoroughly examine the models' performance.

6.1 Scenario A

For the unbalanced Scenario A dataset, the models perform within a range of $84-99+\%$ accuracy, overall. The highest-performing model is Decision Tree (modified CART algorithm), which maintains an extremely high accuracy ($>99.9\%$) over all the data files, excluding the 10s timeout data file. However, the DT and kNN algorithms both perform with exceptional accuracy. The DNN models appear to match or nearly match the performance of DT and kNN in all datasets except 10s, however the keras model suffers from lower performance on the 60s timeout data file. This model also demonstrates a noticeably higher variance than the others. For all models, except for ZeroR, there is a clear, substantial drop in performance for the 10s data file. With the unbalanced dataset, ZeroR performs well with very high ($>95\%$) accuracy on the 120s data file and a minimum accuracy of 84.99% on the 15s data file. The models appear to perform slightly better as the timeout value increases. Table 4 presents the classification Accuracy Metrics from the ZeroR, DT, kNN, keras DNN, and fast.ai DNN models across all five timeout values and datasets for unbalanced Scenario A, presented with one standard deviation of error from the mean classification accuracy across ten folds.

Table 5 displays classification Accuracy Metrics from the ZeroR, DT, kNN, keras DNN, and fast.ai DNN models across all five timeout values and datasets for balanced Scenario A, presented with one

Table 4: Classification Accuracies on Unbalanced Scenario A

File	ZeroR (%)	Decision Tree (%)	kNN (%)	Keras (%)	FastAI (%)
0s	88.14 ± 0.01	95.45 ± 0.34	95.14 ± 0.32	88.14 ± 0.01	89.75 ± 1.82
15s	84.99 ± 0.02	99.91 ± 0.06	99.83± 0.10	99.97 ± 0.04	99.32 ± 0.25
30s	89.22 ± 0.02	99.88 ± 0.09	99.87 ± 0.07	99.80 ± 0.42	99.76±0.07
60s	94.44 ± 0.03	99.94 ± 0.05	99.83 ± 0.09	95.50 ± 2.20	99.86±0.05
120s	95.82 ± 0.00	99.88±0.10	99.92±0.11	99.98±0.06	99.93±0.05

Table 5: Classification Accuracies on Balanced Scenario A

File	ZeroR (%)	Decision Tree (%)	kNN (%)	Keras (%)	FastAI (%)
0s	49.98 ± 0.01	91.27 ± 0.43	91.21±0.57 0.32	80.58±7.87	89.29±0.51
15s	49.94±0.03	99.74±0.23	99.55±0.25	99.88±0.19	98.61±0.54
30s	49.97±0.06	99.60±0.36	99.75±0.21	99.46±0.71	98.89±0.44
60s	49.78±0.12	99.51±0.31	98.96±0.79	94.64±1.64	99.04±0.44
120s	50.00±0.00	99.47±0.90	99.47±0.90	99.36±0.74	99.93±0.05

standard deviation of error from the mean classification accuracy across ten folds. Accuracy results (Table 5 for the balanced version of the Scenario A dataset, there are some substantial differences compared to the same from the unbalanced dataset (Table 4). First, the Keras DNN model has an extremely difficult time with the 10s dataset. This is seen in the fact that the DNN has much higher standard deviation than any other model in this experiment. The baseline model maintains an accuracy around 50%, which is expected since each of the datasets is balanced. Similar to the previous experiment, DT and kNN perform similarly and the DNNs either match accuracy or underperform. Outside of the performance drop with the 10s dataset, there doesn't appear to be a trend of accuracy with the timeout values in this experiment.

6.2 Scenario B

As the timeout values from the dataset increase, there is a near-universal drop in performance for the models (see Table 6). The only model that does not match this trend is the fast.ai DNN model. In contrast to the previous Scenario A experiments, there are higher standard deviation values and lower accuracies. The best-performing model is RF, however DT shows accuracies that are only a few percentage points lower. Table 6 demonstrates the classification Accuracy Metrics for the RF, DT, kNN, keras DNN, and fast.ai DNN models across all five timeout values and datasets for Scenario B, presented with one standard deviation of error from the mean classification accuracy across ten folds.

Table 6: Classification Accuracy Metrics on Scenario B

File	ZeroR (%)	Decision Tree (%)	kNN (%)	Keras (%)	FastAI (%)
0s	81.99±1.25	77.61±0.87	72.87±1.00	72.61±1.55	40.21±1.85
15s	82.02±1.58	78.36±2.18	71.61±2.06	73.78±1.59	67.44±4.62
30s	80.81±1.97	77.31±2.23	67.50±2.57	71.10±2.93	61.16±12.29
60s	78.22±4.47	74.46±4.77	62.73±4.79	61.97±3.91	60.94±1.61
120s	75.74±3.36	72.66±5.16	59.67±6.26	58.63±6.28	56.84±2.93

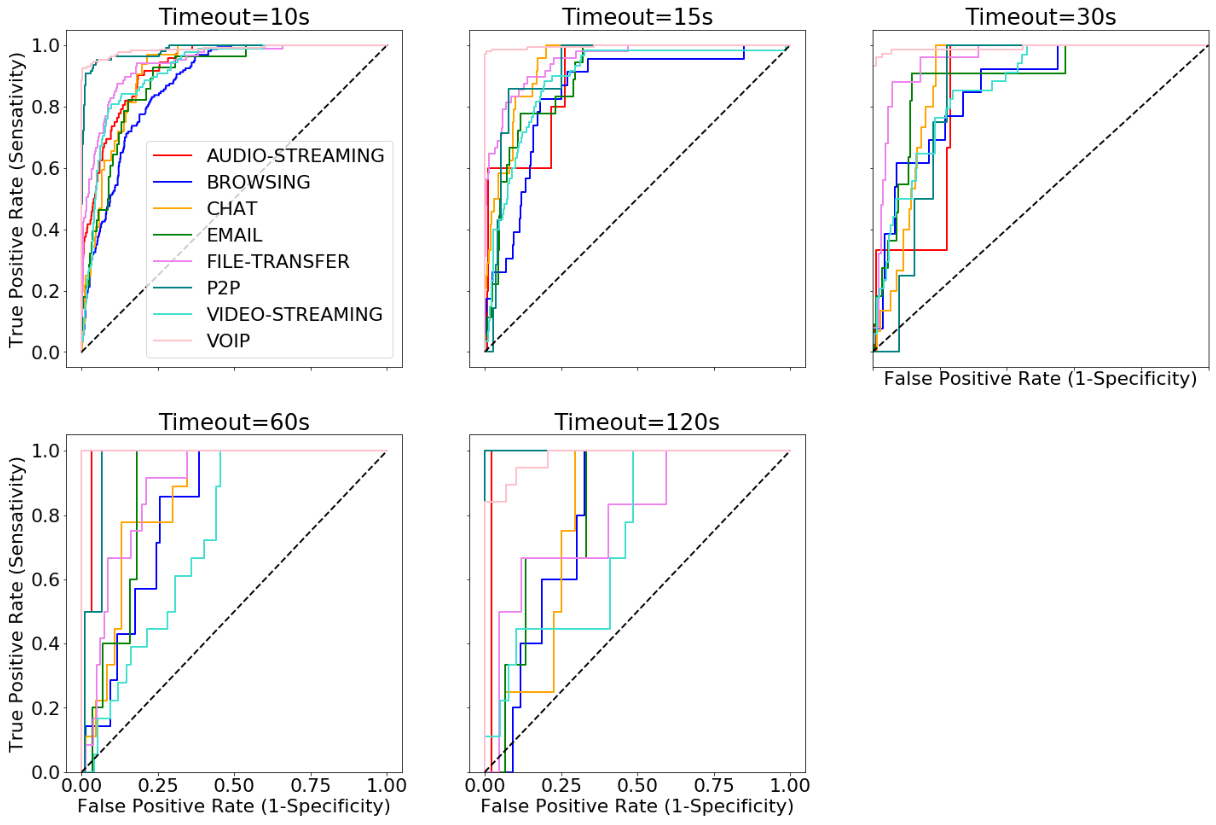

Figure 4: Classification ROC Curves for Scenario B

6.3 ROC Curves

This section will focus on presenting the results from the ROC curves for each dataset. The two types of ROC curves shown below are the ROC of the classes and ROC of the micro- and macro-averages. The class-based ROC graphs contain all the ROC curves for all the classes, while the averages-based ROC graphs show all the averages as discussed in the metrics section. As a recap, the micro-average calculates F1-scores of a model by accounting for the classification frequency, while the macro-average calculates the mean F1-score of all the classes, regardless of the class frequency.

6.4 ROC Curves of the Classes

One trend that is immediately noticeable in the ROC of the classes graphs is the downward trend in AUC for almost all classes as the timeout value increases. While the AUCs for all classifications are > 0.90 for the 10s dataset, most of these classes' AUCs decrease to 0.75 for the 120s dataset. However, the models appear to consistently perform extremely well when classifying for the Audio-Streaming, P2P, and VoIP classifications. The area under the curve for the VoIP is reported the highest (> 0.99) for all timeout values. While Audio-Streaming and P2P classifications are the most underrepresented in the datasets, they appear to show some of the highest performance metrics, with both AUC's ranging $0.90 - 1.00$. In sharp contrast, despite the fact that Video-Streaming - which reaches an AUC high of 0.91 and low of 0.68 - is one of the majority classifications in most of the datasets, it is one of the most poorly classified categories in the dataset. Figure 4 represents various ROC Curves for the Classes across all five datasets with varying timeout values.

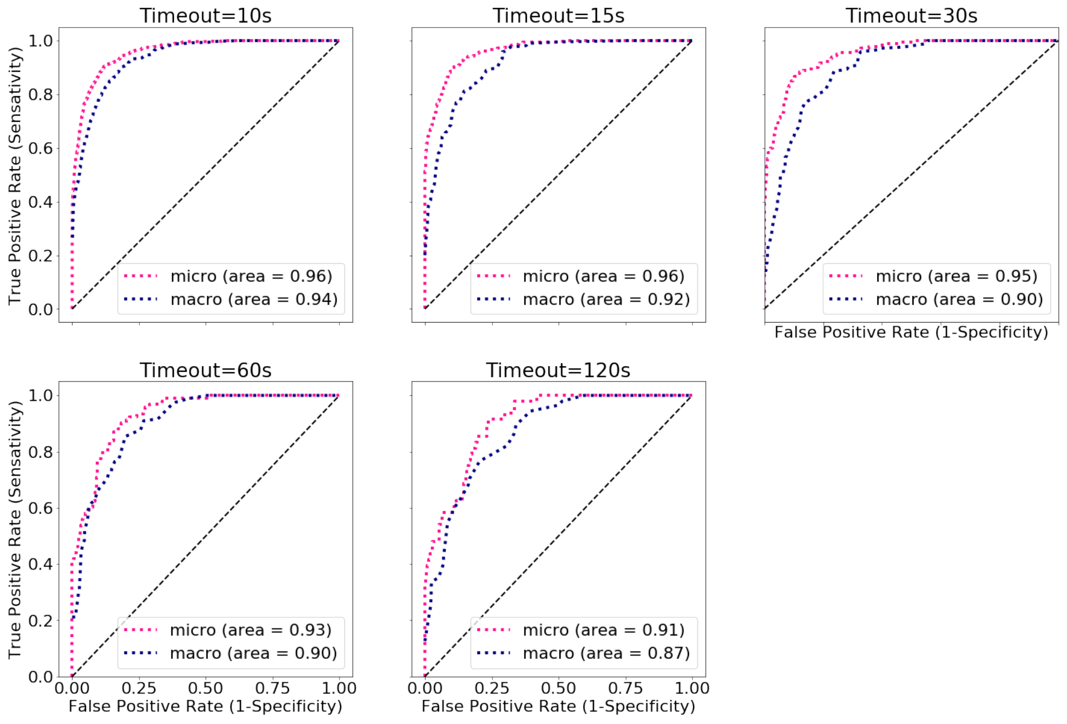

Figure 5: Micro- and Macro-Average ROC Curves for Scenario B

6.5 ROC of the Micro- and Macro-Averages

The micro-average and macro-average ROC curves decrease with the increase in timeout value. This result is expected since the micro-average and macro-average both take the recall and precision metrics for each classification into account, as discussed above. The micro-average's AUC has a maximum of 0.96 and decreases to 0.89 for the 120s timeout dataset. Starting from a high of 0.94, the macro-average's AUC decreases to 0.84; thus, it remains only slightly lower than the micro-average's AUC result across all five datasets. Receiver Operating Characteristic Curves for the Micro- and Macro-Averages across all five datasets with varying timeout values are presented in Figure 5. Area under the Curve metrics presented in the legend.

6.6 Confusion Matrices

Figure 6 represents various confusion matrices from Scenario B experiments with varying timeout values from all five datasets. The numbers presented are from the final fold of each model's testing. There is a clear decrease in the size of the datasets as the timeout values increase, potentially reducing the quality of interpretations. It can be clearly observed that the dataset's quality quickly decreases as the timeout values increase and the models over-classify for the Video-Streaming category. Despite the fact that the VoIP classification is far more prevalent than most of the other classes in the datasets - to the extreme of the dataset containing 4 times more VoIP than any other class - the models never over-classify VoIP. This will be explored further in the Discussion section.

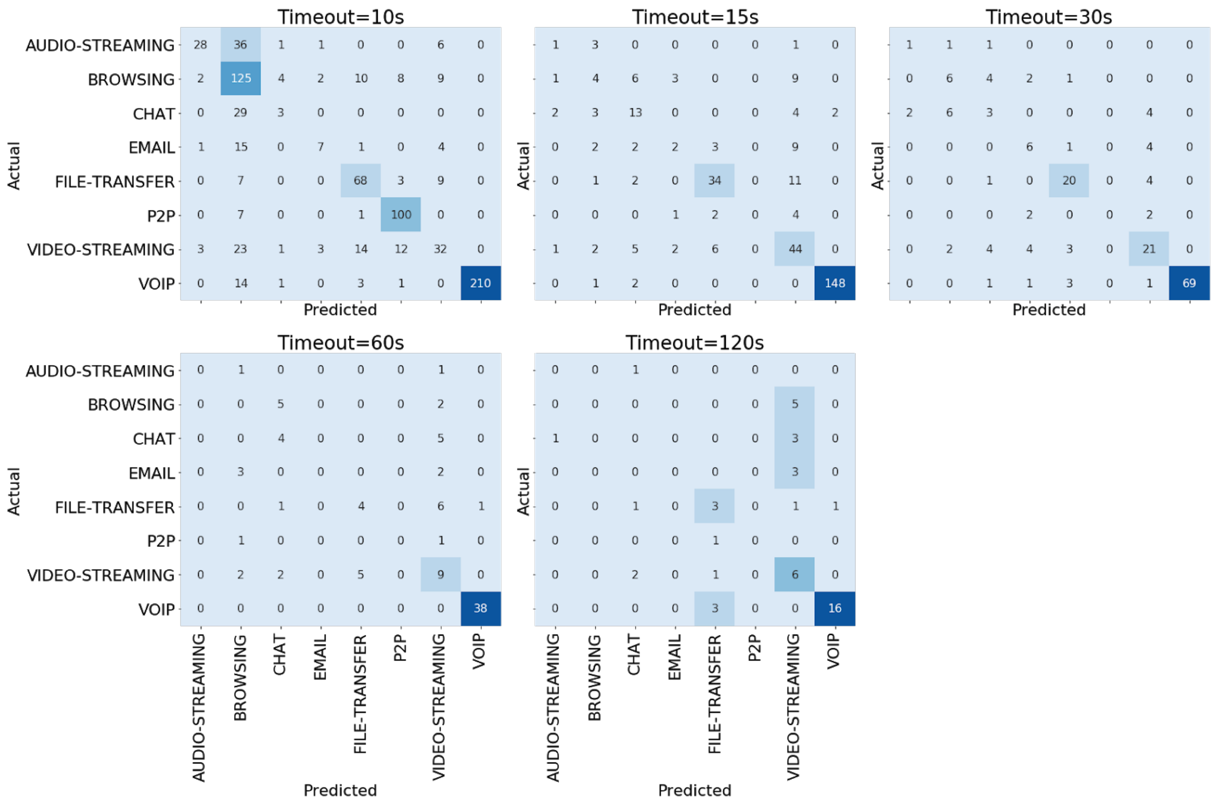

Figure 6: Confusion Matrices for Scenario B

7 Discussion

In this section, we attempt to further discuss the results described in the previous section, intuitively demonstrating why the models from Scenario A perform extremely well while those from Scenario B seemingly lack enough diversity within the data to accurately classify $> 40\%$ of testing samples. The distributions of the features analysed, in addition to the classifications, are graphed in violin plots in Figure 7. Almost all models trained on the balanced and unbalanced Scenario A datasets perform with high accuracy of more than 98% except for the 10s latency dataset which still attained decent accuracy results of little over 88%. We investigated the characteristics of the features in Scenario A to understand and explain the discrepancy in the result. The 10s dataset's poorer results follow along with the observation made by Lashkari et al. [26] where they noted that the lower latency datasets were harder to distinguish than the higher latency datasets.

The features chosen by the UNB-CIC work are selected for demonstration and each feature's value and classification distribution are presented below. The Forward IAT Min (min_fiat), Backward IAT Mean (mean_biat), and Backward IAT Total (total_biat) features generally show that non-Tor samples are prone to lower values, however showing wider distributions overall. While the Backward IAT Max feature distribution shows similar characteristics, it is to a lesser extent. This result strengthens the research seen in [38] that the binary classification of Tor and nonTor packets is trivial given time-based datasets.

Figure 7 shows various violin plots of four features from the Scenario A dataset across five timeout values. These graphs show the probability distribution of the classification and values from each feature. Note that the 'total_biat' feature is unavailable in the 60s and 120s datasets.

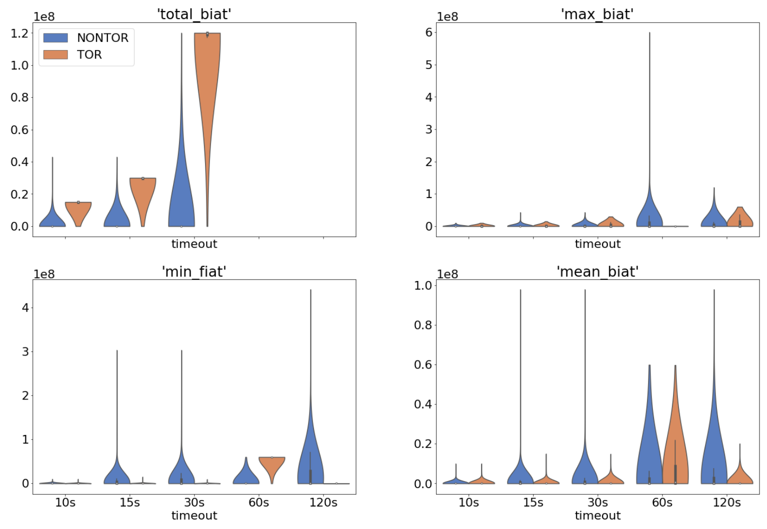

Figure 7: Scenario A Feature Class Distributions on Violin Plots

The only model trained on Scenario A that did not perform well was ZeroR with a maximum classification accuracy of 95.82% and 50.00% for the unbalanced and balanced datasets, respectively. These results are expected and direct results of the datasets' composition. Since ZeroR is a simple classifier that always predicts the majority class in the dataset composition, there is a clear pattern seen in the results. In the unbalanced Scenario A dataset, for example, 95% of the data is labelled as nonTor. The balanced Scenario A dataset is, by definition, a 50-50 split of the nonTor and Tor samples. Thus, the highest performance possible with stratified k-fold cross validation is 50%.

The models trained on Scenario B demonstrate mixed classification performance. As an ensemble method of decision trees, RF typically outperforms DT. K-Nearest Neighbor performs moderately well, considering there are only eight classes. Our experiments show that kNN performs worse than both DT and RF despite the literature indicating otherwise in various settings [37][35][16], due to the colloquial "curse of dimensionality" [15][39]. The dimensionality problem indicates that, as the dimensions of a problem increase, the size of the training dataset should also increase, exponentially. Various works also show similarity in the performance of kNN and DNN models [21][19]. These findings aligns with the results presented within this work. The results from the fast.ai model are potentially a consequence of poor data quality in the 10s dataset in conjunction with an overly-complex model. While the fast.ai model performance on the testing set of the 10s dataset is extremely low compared to RF, DT, kNN, and keras DNN models, classification accuracy is similar to the keras DNN model for the remaining datasets, as expected.

For Scenario B, the graphs of the distribution of the features (Figure 8) clearly shows the source of the difficulty the models are experiencing when classifying across all eight categories, however demonstrating the reasons for specific classifications (Audio-Streaming and VoIP) exceeding the mean performance metrics.

The distribution of the Audio-Streaming class is non-conforming to most of the other classes in the dataset. In each violin plot in Figure 8, this class can clearly be differentiated from the other categories due to its generally wider distribution and - as seen in the duration feature - its lower values. The VoIP class shows unique distribution behavior as well. However, this is derived from its consistently low values and short distribution width.

From these graphs, multiple classifications, such as Browsing, Chat, and Video-Streaming, do not stand out enough to classify, intuitively. While these models exceed the performance of human-based

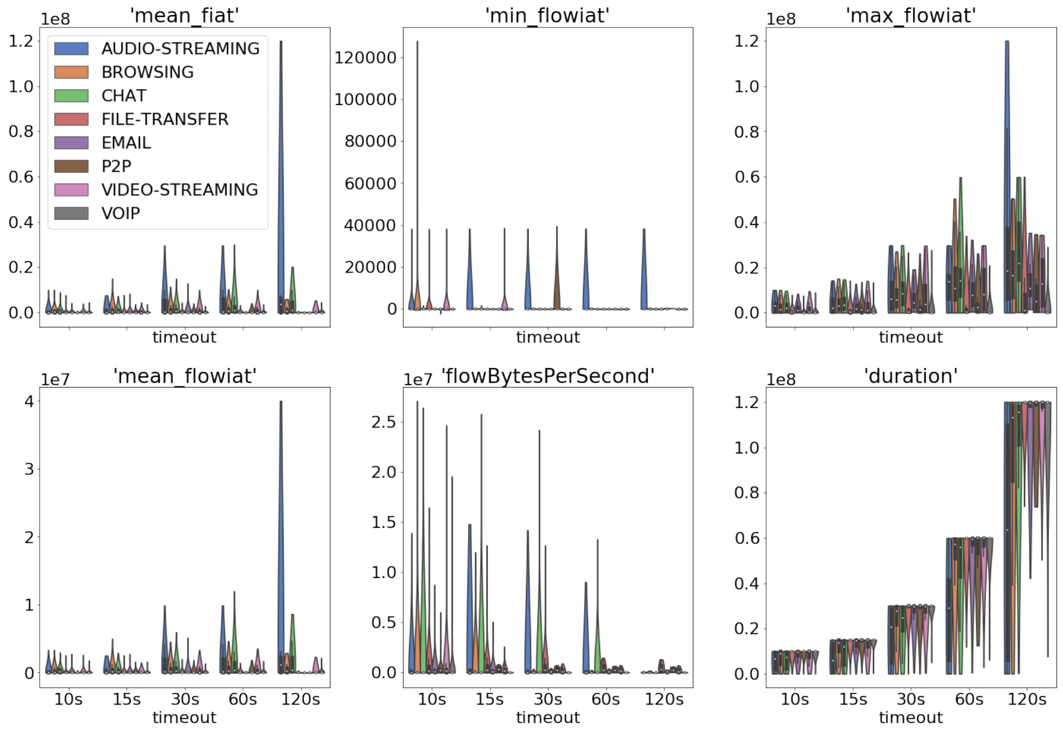

Figure 8: Scenario B Feature Class Distributions on Violin Plots

classification, these distributions, and low data quantities, provide an insight into the low performance of some of the models.

Figure 8 shows various violin plots of six features from the Scenario B dataset across five different timeout values. These graphs show the probability distribution of the classification and values from each feature. The classifications are each graphed from left to right as follows: Audio-Streaming, Browsing, Chat, File-Transfer, Email, Peer-to-Peer, Video-Streaming, and VoIP, as indicated on the legend.

While Audio-Streaming, P2P, and Video-Streaming have high AUCs, most of the datasets in Scenario B suffer from extremely imbalanced classes and overall low counts of samples. We attribute the decrease in the quality of the models to the decrease in size of the datasets as the timeout values increase. An important consequence of this is clearly shown in Scenario B's composition, with only a few samples for most of the classifications present. For the 120s Scenario B dataset, the five classifications with the lowest count, 63% of the available classifications, take up only 29% of the samples. Similarly, the top two majority classifications (25% of the classes) contain 58% of the samples. Considering how the datasets were collected and generated [26], this seems to be an unavoidable issue and the performance drop is expected with all models.

8 Conclusion and Future Work

This work analyzed the performance of four machine learning algorithms (ZeroR, Random Forest, CART, and k-Nearest Neighbors) and two deep learning frameworks (Keras and fast.ai) on the ISCX-Tor2016 datasets for two different scenarios. Scenario A classifies Tor from non-Tor traffic with a very high accuracy of more than 99%. The experiment results from Scenario A align with prior work in that the differentiation between Tor and nonTor network traffic is trivial using only time-based features. All six learning models performed with $> 99\%$ accuracy, even with the balanced dataset.

The classification of various Tor traffic presented in Scenario B dataset is still an unsolved problem. RF and DT classifiers demonstrated superior performance in Scenario B, moreover increased data features such as the amount of packets in a flow and other statistics may improve performance for all models.

While Tor traffic is resistant to classification, third parties are able to easily detect Tor traffic, potentially endangering users. Experimental results show that if the Tor Project intends on protecting users from external entities, then more effort must be expended on reducing the time-based fingerprints left by the Tor protocol. Additional research could more systematically investigate the features within the ISCXTor2016 dataset and experiment with other successful deep learning models such as the convolutional neural network.

Acknowledgment

This research project was supported by the state of Colorado through funds appropriated for cybersecurity law dubbed "Cyber Coding Cryptology for State Records." Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding sources.

References

[1] fast.ai — about. `https://www.fast.ai/about` [Online; accessed on July 29, 2020].

[2] Keras — about keras. `https://keras.io/about/` [Online; accessed on August 5 2020].

[3] Scikit-learn - 1.10. decision trees. =https://scikit-learn.org/stable/modules/tree.html [Online; accessed on July 12, 2020].

[4] Scikit-learn - 1.11 ensemble methods. `https://scikit-learn.org/stable/modules/ensemble.html` [Online; accessed on July. 29 2020].

[5] University of new brunswick canadian institute for cybersecurity — applications. `https://www.unb.ca/cic/research/applications.html` [Online; accessed on July 18, 2020].

[6] Weka 3 - data mining with open source machine learning software in java. `https://www.cs.waikato.ac.nz/ml/weka/` [Online; accessed on July 30, 2020].

[7] Whonix - design and goals. `https://www.whonix.org/wiki/About` [Online; accessed on August 5, 2020].

[8] Tor project, "udp over tor," tor project, 2012. `https://trac.torproject.org/projects/tor/ticket/7830` [Online; accessed on July 18, 2020], 2012.

[9] fastai. `https://docs.fast.ai/` [Online; accessed on August 29 2019], 2019.

[10] Tor project — anonymity online. `https://www.torproject.org/` [Online; accessed on July 15, 2020], 2020.

[11] K. Abe and S. Goto. Fingerprinting attack on tor anonymity using deep learning. In *Proc. of the 13th APAN Research Workshop (APAN'16), Hong Kong, China*, pages 15–20. Asia-Pacific Advanced Network, March 2016.

[12] M. Aminuddin, Z. Zaaba, M. Singh, and D. Sing. A survey on tor encrypted traffic monitoring. *International Journal of Advanced Computer Science and Applications*, 9(8), 2018.

[13] R. B. Basnet, R. Shash, C. Johnson, and T. Doleck. Towards detecting and classifying network intrusion traffic using deep learning frameworks. *Journal of Internet Services and Information Security (JISIS)*, 9(4):1–17, 2019.

[14] K. Bauer, M. Sherr, D. McCoy, and D. Grunwald. Experimentor: A testbed for safe and realistic tor experimentation. In *Proc. of the 4th Workshop on Cyber Security Experimentation and Test (CSET'11), San Francisco, California, USA*. USENIX Association, August 2011.

[15] R. Bellman. *Dynamic Programming*. Dover Publications, 2003.

[16] A. Bilal, W. Jian, and M. Shafiq. Intrusion detection by using hybrid of decision tree and k-nearest neighbor. *International Journal of Hybrid Information Technology*, 9(12):201–208, December 2016.

[17] L. Breiman. Random forests. *Machine Learning*, 45:5–32, October 2001.

[18] L. Breiman and P. Spector. Submodel selection and evaluation in regression - the x-random case. *International Statistical Review / Revue Internationale de Statistique*, 60(3):291–319, December 1992.

[19] B. Bromley. Neural-network and k-nearest neighbor classifiers. Technical report, AT&T Bell Laboratories, August 1991.

[20] A. Chaabane, P. Manils, and M. A. Kaafar. Digging into anonymous traffic: A deep analysis of the tor anonymizing network, September 2010.

[21] A. Eskanadarinia, H. Nazapour, M. Teimouri, and M. Ahmadi. Comparison of neural network and k-nearest neighbor methods in daily flow forecasting. *Journal of Applied Sciences*, 10(11):1006–1010, 2010.

[22] H. J. Jeong, W. Hyun, J. Lim, and I. You. Anomaly teletraffic intrusion detection systems on hadoop-based platforms: A survey of some problems and solutions. In *Proc. of the 2012 15th International Conference on Network-Based Information Systems (NBiS'12), Melbourne, Victoria, Australia*, pages 766–770. IEEE, September 2012.

[23] C. Johnson, B. Khadka, R. B. Basnet, and T. Doleck. Towards detecting and classifying malicious urls using deep learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 11(4):31–48, 2020.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. `https://arxiv.org/abs/1412.6980v9` [Online; accessed on July 13, 2020], 2015.

[25] R. Kohavi. "a study of cross-validation and bootstrap for accuracy estimation and model selection". In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, Quebec, Canada*, page 1137–1143. Morgan Kaufmann Publishers Inc., August 1995.

[26] A. H. Lashkari, D. G. Gil, M. Mamun, and A. Ghorbani. Characterization of tor traffic using time based features. In *Proc. of the 3rd International Conference on Information Systems Security and Privacy (ICISSP'17), Porto, Portugal*, pages 253–262. SciTePress, February 2017.

[27] Z. Ling, J. Luo, K. Wu, W. Yu, , and X. Fu. Torward: Discovery of malicious traffic over tor. In *Proc. of the 2014 IEEE Conference on Computer Communications (INFOCOM'14), Toronto, Ontario, Canada*, pages 1402–1410. IEEE, April-May 2014.

[28] Y. Liu, S. Li, C. Zhang, C. Zheng, Y. Sun, and Q. Liu. Itp-knn: Encrypted video flow identification based on the intermittent traffic pattern of video and k-nearest neighbors classification. In *Proc. of the 20th International Conference on Computational Science (ICCS'20), Amsterdam, The Netherlands*, volume 12138 of *Lecture Notes in Computer Science*, pages 279–293. Springer, Cham, June 2020.

[29] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Comparison of trends in practice and research for deep learning. `https://arxiv.org/abs/1811.03378` [Online; accessed on July 29, 2019], 2019.

[30] J. Opitz and S. Burst. Macro f1 and macro f1,. `https://arxiv.org/abs/1911.03347v2` [Online; accessed on July 12 2020], 2019.

[31] B. Park, J. W.-K. Hong, and Y. J. Won. Toward fine-grained traffic classification. *IEEE Communications Magazine*, 49(7):104–111, July 2011.

[32] B. Park, Y. Won, M. Kim, and J. Hong. Towards automated application signature generation for traffic identification. In *Proc. of the 2008 IEEE Network Operations and Management Symposium (NOMS'08), Salvador, Brazil*, pages 160–167. IEEE, April 2018.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85), 2011.

[34] N. M. R. Dingledine and P. Syverson. Tor: The second-generation onion router. `https://pdfs.semanticscholar.org/bdc8/7add980372e7cf8c27e1af47d7ce282092c8.pdf` [Online; accessed on July 13, 2020], 2014.

[35] H. Rajaguru and S. Chakravarthy. Analysis of decision tree and k-nearest neighbor algorithm in the classifi-

cation of breast cancer. *Asian Pacific Journal of Cancer Prevention*, 20(12):3777–3781, December 2019.

[36] Z. Rao, W. Niu, X. Zhang, and H. Li. Tor anonymous traffic identification based on gravitational clustering. *Peer-to-Peer Networking and Applications*, pages 592–601, June 2017.

[37] T. Saragih, D. Fajri, and A. Rakhmandasari. Comparative study of decision tree, k-nearest neighbor, and modified k-nearest neighbor on jatropha curcas plant disease identification. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, 5(1):55–60, February 2020.

[38] M. Soleimani, M. Mansoorizadeh, and M. Nassiri. Real-time identification of three tor pluggable transports using machine learning techniques. *The Journal of Supercomputing*, 74:4910–4927, February 2018.

[39] C. Taylor. *Applications of Dynamic Programming to Agricultural Decision Problems*. CRC Press, 2019.

Author Biography

Clayton Johnsongraduated from Colorado Mesa University (CMU) with a BS in Computer Science and a Professional Certificate in Cybersecurity in 2020. He's currently pursuing his PhD in Cybersecurity at the University of Colorado Boulder.

Bishal Khadka is a senior undergraduate student pursuing his Bachelor's in Computer Science and Professional Certificate in Cybersecurity degrees at Colorado Mesa University (CMU). Bishal is currently the president of Cybersecurity club and a research fellow at the Cybersecurity Center at CMU.

Ethan Ruiz graduated with a Bachelor's in Computer Science and a Professional Certificate in Cybersecurity from Colorado Mesa University (CMU) in 2020. He is the former treasurer of the Cyber Security Club at CMU.

James Halladay is a junior undergraduate pursuing his Bachelor's in Math and Computer Science at Colorado Mesa University. His research interests are in Complex Analysis, Machine Learning, Front-end Design, and Graph Theory.

Tenzin Doleck received his PhD from McGill University in 2017. He is a Canada Research Chair and Assistant Professor at Simon Fraser University.

Ram B. Basnet is an associate professor of Computer Science and Cybersecurity at Colorado Mesa University (CMU). He received his BS in Computer Science from CMU in 2004 and MS and PhD in Computer Science from New Mexico Tech in 2008 and 2012, respectively. His research interests are in the areas of information assurance, machine learning, and computer science pedagogy.