2021-09

TOWARDS VIDEO FINGERPRINTING ATTACKS OVER TOR

Campuzano, Carlos D.

Monterey, CA; Naval Postgraduate School

http://hdl.handle.net/10945/68304

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TOWARDS VIDEO FINGERPRINTING ATTACKS OVER TOR

by

Carlos D. Campuzano

September 2021

Thesis Advisor:	Armon C. Barton
Second Reader:	Vinnie Monaco

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE TOWARDS VIDEO FINGERPRINTING ATTACKS OVER TOR	**5. FUNDING NUMBERS**
6. AUTHOR(S) Carlos D. Campuzano	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

As web users resort to adopting encrypted networks like Tor to protect their anonymity online, adversaries find new ways to collect their private information. Since videos over the internet are a major source of recruitment, training, incitement to commit acts of terrorism, and more, this project envisions developing a machine learning algorithm that can help the Department of Defense find terrorists who take advantage of the dark web to help promote extremist ideology. This thesis describes the steps for training a machine learning classifier in a closed-world scenario to predict YouTube video patterns over an encrypted network like Tor. Our results suggest an adversary may predict the video that a user downloads over Tor with up to 92% accuracy, or may predict the length of a video with error as low as 5.3s. Similar to known website fingerprinting attacks, we show that Tor is susceptible to video fingerprinting, suggesting that Tor does not provide the level of anonymity as previously thought.

14. SUBJECT TERMS Tor, website fingerprinting, video fingerprinting, anonymity, machine learning classifier, deep learning, dark web			**15. NUMBER OF PAGES** 59
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

TOWARDS VIDEO FINGERPRINTING ATTACKS OVER TOR

Carlos D. Campuzano
Lieutenant Commander, United States Navy
BS, State University of New York Empire State College, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

**NAVAL POSTGRADUATE SCHOOL
September 2021**

Approved by: Armon C. Barton
Advisor

Vinnie Monaco
Second Reader

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As web users resort to adopting encrypted networks like Tor to protect their anonymity online, adversaries find new ways to collect their private information. Since videos over the internet are a major source of recruitment, training, incitement to commit acts of terrorism, and more, this project envisions developing a machine learning algorithm that can help the Department of Defense find terrorists who take advantage of the dark web to help promote extremist ideology. This thesis describes the steps for training a machine learning classifier in a closed-world scenario to predict YouTube video patterns over an encrypted network like Tor. Our results suggest an adversary may predict the video that a user downloads over Tor with up to 92% accuracy, or may predict the length of a video with error as low as 5.3s. Similar to known website fingerprinting attacks, we show that Tor is susceptible to video fingerprinting, suggesting that Tor does not provide the level of anonymity as previously thought.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AL	Adaptation Logic
API	Application Programming Interface
CNN	Convolutional Neural Network
DASH	Dynamic Adaptive Streaming over HTTP
DF	Deep Fingerprinting
DL	Deep Learning
DoD	Department of Defense
DTC	Decision Tree Classifier
GBR	Gradient Boost Regressor
ISP	Internet Service Provider
kNR	K-Neighbors Regressor
kNN	K-Nearest Neighbor
LRC	Logistic Regression Classifier
LSTM	Long Short-Term Memory
MBR	Multi Bit Rate
ML	Machine Learning
MAE	Mean Absolute Error
MSE	Mean Squared Error
NPS	Naval Postgraduate School

PCAP	Packet Capture
QoE	Quality of Experience
RBF	Radial Basis Function
RFC	Random Forest Classifier
RFR	Random Forest Regression
RMSE	Root Mean Squared Error
SDAE	Stacked Denoising Autoencoder
SML	Supervised Machine Learning
SVM	Support Vector Machines
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TBB	Tor Browser Bundle
Tor	The Onion Router
VF	Video Fingerprinting
VM	Virtual Machine
W-T	Walkie-Talkie
WF	Website Fingerprinting
WTF-PAD	Website Traffic Fingerprinting Protection with Adaptive Defense

Acknowledgments

I want to thank my thesis advisor, Dr. Armon Barton, for his patience in teaching me about machine learning and for his input in the writing and completion process. I would also like to thank Dr. John Monaco as my second reader for his good advice. To my friend Alvaro Mancero for his support in those moments when I needed it. To my parents for their endless support. And lastly, to my wife, Liliana, and my daughters, Fiorella and Mia. Thank you for your love and patience, and for being part of this journey by giving me the strength I needed to move forward.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

1.1 Introduction

As the Department of Defense (DoD) and other agencies move to develop and integrate Machine Learning (ML) strategies to fight terrorists who use the dark web for criminal activities, video training and recruiting capability continues to be a valuable tool used by terrorists [1]. While computer technology continues to advance, anonymous communications has become an enabler for criminal behavior in the dark web [2]. However, it has been shown that some forms of traffic analysis may be used to predict people's behavior on the dark web. Such findings suggest that the dark web is not as anonymous as previously thought [3].

The objective of this research is to identify video traffic patterns over The Onion Router (Tor) protocol. By impersonating an adversary, we first build a database in which we passively observe video traffic between a client and her Tor connection. Then, we train various classifier and regressor ML models to identify traffic patterns. Using this attack model, we predict the name and length of a video that users are watching over Tor. The attack is called Video Fingerprinting (VF). This research aims to use an ML classifier that can achieve high accuracy on predicting a user's video download over an encrypted network such as Tor. Our findings support that Tor may not be as private as once thought. Moreover, VF attacks performed on law-abiding users may prove to be a privacy challenge that must be taken into consideration.

The focus of this research is to asses a Tor adversary's ability to disrupt illegal recruitment and training of terrorist organizations. This will be completed by determining the subject of a video a terrorist sympathizer watches over the Tor network. More specifically, this thesis explores how an ML classifier can be trained in a supervised environment to predict, with high confidence, videos by observing encrypted traffic patterns over Tor.

1.2 Purpose and Scope

The primary focus of this research is to create a VF attack model by collecting a high number of video captures that can be used to train a detection algorithm for predicting video patterns using ML techniques. The ML process will create models used to detect unusual frequent patterns of videos in a VF attack environment. The goal is to study which traffic predictors correlate best for video prediction. It is also important to evaluate how different ML models perform in these various attack settings. We expect that the VF attack may reach similar results compared to Website Fingerprinting (WF) attacks given that similar features are used.

One hypothesis is that ML classifiers trained on encrypted video streams may be used by adversaries to predict which videos users are watching. However, we seek to learn if ML can in fact be used to predict user videos over encrypted traffic, and which ML models perform best for predicting various labels such as a unique video identifier and a video length. This research also focuses on learning if video streaming times correlate well enough with video length to be used as a predictor for improving accuracy.

In addition, this research explores Deep Learning techniques proposed by Rahman et al. [4] for WF attacks. We achieved similar results when we run Sirinam et al.'s Deep Fingerprinting models [5] to perform the VF attack against Tor.

1.3 Thesis Organization

The thesis is organized as follows. Chapter 2 is an overview of the Tor network, previous research on WF attack models, and provides a description of how this thesis proposes to use the same WF concept to perform the VF attack over Tor. It also includes a brief description of the different classifier and regressor ML algorithms that will be used in the thesis. Chapter 3 describes our methodology including our process for obtaining the data using a `web crawler` program, our data representation for exploration, our assumptions, and our attack models. Chapter 4 describes how the data was generated in preparation to be used for exploration with the different ML algorithms. It describes the results of our experiments and major findings. Chapter 5 provides recommendations for future work.

CHAPTER 2:
Background and Related Work

This chapter provides a comprehensive review of the literature related to the Tor network and its challenges against WF attacks. It explores different studies that aim to predict the websites a user visits over Tor by passively analyzing the traffic between the user and her Tor connection. A similar approach may be used for predicting videos over Tor, which is the scope of the rest of this thesis.

2.1 Networking Overview

As millions of people continue to look for ways to keep their identity private and protected from passive adversaries, the demand for anonymous communication systems has significantly increased in recent years. Standard requests for information on the Internet use Transmission Control Protocol (TCP), a highly symmetric protocol in that a client and destination can transmit and receive data packets simultaneously [6]. In addition, the data packet is encrypted as part of the Transport Layer Security (TLS) handshake to ensure that sensitive information cannot be seen by eavesdroppers [7].

However, a passive adversary may still profile users by observing their URLs or the websites they routinely visit [8]. One popular system is called Tor which allows users to build an encrypted channel using randomly chosen proxies before making any requests on the Internet [9]. This distributed architecture makes it very difficult for a passive adversary to link a client to her destination, unless the adversary can observe traffic between both the client and her entry node, and simultaneously the exit node and her destination; in which case the adversary may deanonymize the client by performing an end-to-end timing attack [10]. Additionally, other research has shown that the privacy Tor offers has been challenged by passive and active attacks [11].

2.2 Tor Onion Router

Tor is an open-source network that allows anonymous communication by not letting an eavesdropper link a client to her destination. To establish a *secure connection*, Tor chooses a path to build an encrypted *circuit* in which it routes encrypted data between a client and destination through a random selection of three Tor relay nodes.

Shown in Figure 2.1 is a graphical representation of a Tor circuit. By default, the client establishes a set of circuits and stores them in the circuit pool to be ready for use. When client (A) makes a connection request to reach destination (B), a ready made circuit is retrieved from the circuit pool. The client then receives a random circuit identification number that is time stamped. After ten minutes, the circuit is switched to a new circuit that is waiting in the circuit pool. The same process continues for the duration of the session. Through one circuit, a client may connect to several different destination websites because each circuit can be shared by several TCP *streams* [12].

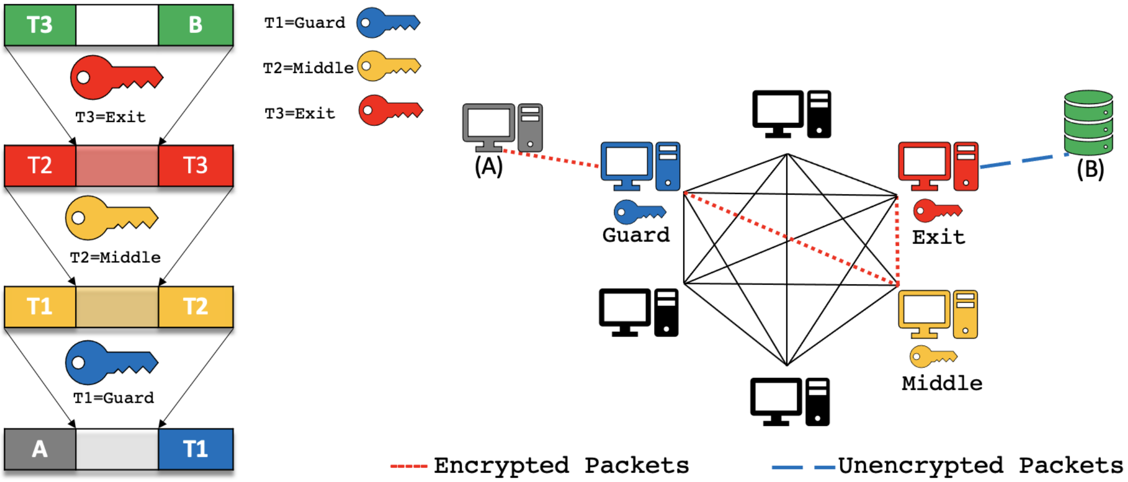


Figure 2.1. Tor network.

The first hop in a Tor circuit is known as the *Guard* node. By establishing a unique symmetric key between the *Guard*, *Middle*, and *Exit* nodes, the client can route encrypted data. Initially, the client is in possession of all three keys. The Tor browser encrypts the packet with the three keys on top of the other in reverse order. As the *Guard* node T1 receives the packet, it strips off the address of the *Middle* node T2. The *Middle* node T2 follows the same step, it decrypts the packet and forwards the packet onto the *Exit* node T3. Finally, the *Exit* node decrypts the packet and determines the destination. Each of these nodes knows its predecessor and successor are but no other node in the circuit. The final destination receives a data packet without a Tor encrypted layer or the identity of the client. Instead, it is given the identity of the Tor *Exit* node, in this example, T3.

In attempts to reveal possible attacks on the Tor network, researchers have experimented with ways to test the anonymity of the network against traffic analysis. Danezis and Murdoch [13], presented a technique on how an adversary has the advantage to collect and analyze the routing of TCP streams over the Tor network between a client and the *Guard* node. Although they point out in their research that their results did not allow them to predict the origin of the client, they were able to uncover visited websites over a Tor connection [14]. In addition, it was noted that the timing characteristics of a stream on a Tor node do not change significantly, and the volume of traffic carried in one stream influences the latency of other streams [13].

2.3 Website Fingerprinting

Recent studies have shown that Tor is vulnerable to WF attacks. Herrmann et al. [8], describe how WF allows an adversary to passively identify websites in an encrypted connection by analyzing the patterns in the network. Their classifier was able to identify 97% of 775 websites over Tor. Pachenko et al. [15] describes how an adversary with robust capabilities as the Internet Service Provider (ISP) or limited resources from a home WiFi connection could eavesdrop on a traffic transmission between a client and a Tor guard node. By extracting features such as time, packet length, and direction without breaking the encrypted transmission, the adversary could create a dataset to train a ML. Their research improved website recognition over Tor from 3% to 55%".

Related work has examined the use of different types of ML classifiers to recognize several websites. In addition, the main page of a website, referred to as the index, has been utilized in the build-up of datasets with the goal to reach a successful WF attack.

Pachenko et al. [16] developed an approach called the *novel WF* attack that begins with the collection of a large dataset consisting of more than 300,000 of the most commonly visited websites. They assume that the adversary has the resources to download and store this large dataset. Using the K-Nearest Neighbor (kNN) classifier approach proposed by Wang et al. [17], they evaluated the characteristics between a set of websites and individual webpages by observing the transmission of metadata. Since the kNN classifier relies on distance, it selects the length of the network packet, timestamp, and the outgoing packets. In addition, collection of data from different locations showed that the accuracy was similar for clients that use the same *network*. The authors point out that their assumptions result in one of the most realistic attacks; they explain that this type of attack is costly for the adversary since a large-scale dataset requires constant review of website updates.

Traditional ML algorithms have proven to be successful for performing WF attacks. However, Sirinam et al. [5], proposed a new type of WF attack called Deep Fingerprinting (DF) that is based on a Convolutional Neural Network (CNN). Since Deep Learning (DL) models do not require selection or fine-tuning of features by hand, they can produce high accuracy on raw network traffic without the need for much prepossessing. The goal of this research was to reevaluate the prior work [16] and [17] by introducing three different DL models: 1) a CNN model, 2) a Long Short-Term Memory (LSTM) model, and 3) a Stacked Denoising Autoencoder (SDAE) model [18]. The results show a high accuracy of 94% for the CNN model when used against Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD) and Walkie-Talkie (W-T) defenses. This suggests that, through automated techniques, adversaries can train DL models to be successful for WF attacks.

Cai et al. [19], proposed an attack on Tor using *pipelining*. *Pipelining* is a sequence of data processing components. Since ML components run asynchronously, the pipeline improves performance by enabling data to be transformed and correlated into a model that can be explored to reach results [20]. Since HTTP requests can be split into multiple partial requests as cover traffic, *pipelining* can analyze the requests simultaneously even if the order of requests are obfuscated. The authors used a Tor version of Firefox to demonstrate the effectiveness of an attack when HTTP requests were made in *random order*. For the test, all packets were padded to ±1500 to obtain the same probability. The results were positive, showing a success of more than 83% with a dataset size of 100 webpages.

2.4 Attack Models

The two attack models that are prevalent for video fingerprinting attacks are called *closed-world* and *open-world*. A closed-world attack assumes the user will only download videos that the adversary used to train their model [21]. This assumption is somewhat relaxed with respect to the adversary's probability of success because the model will never be challenged with videos that it has not seen during training. For example, in our experiments, we use the closed-world model in which the adversary trains on nine videos; during test time, the user may not download any other videos on the Internet except for these nine.

In the real world, it is impossible to train on all the possible videos a user could download. However, to simulate this, we can use an open-world attack model in which the user is allowed to download videos that the adversary did not include in their training set. The adversary would then train their ML algorithm on both videos of interest and a set of other open world videos. At test time, the model will be better equipped for handling unknown samples given that the user is allowed to download videos that the adversary did not include in their training set.

2.5 Video Fingerprinting

In recent years, video streaming has become popular for educating, informing, and entertaining large audiences over the Internet. However, similar to WF attacks on websites, network analysis presents a privacy concern when an adversary uses VF techniques to identify video streams over an encrypted network. Following the same WF concept, an adversary could build a dataset in which they passively observe and download traffic between a client and her Tor guard node [22]. Then, the adversary may train an ML classifier to identify traffic patterns. Using this attack model, the adversary may predict which videos clients are watching over Tor.

To the best of our knowledge, prior research on VF attacks have been done on popular video streaming websites such as Netflix and Chrome [22], [23] among others. Following previous work from Rahman et al. [4], we focused our research on YouTube videos stream over the Tor network. The intent of this research is to explore different ML models and understand how they perform on VF attacks against users who browse YouTube videos over Tor.

Knowing that the number of users who watch videos on the Internet has surged in recent years, Dubin et al. [24], explored a VF attack on HTTP encrypted adaptive streaming by predicting the name of a video. To improve the Quality of Experience (QoE), video streaming uses Dynamic Adaptive Streaming over HTTP (DASH), a Multi Bit Rate (MBR) streaming method where a video is divided into segments and encoded multiple times so that the Adaptation Logic (AL) algorithm may select the best representation of each segment. This work focused on YouTube videos played from a Chrome browser. The authors found that a Support Vector Machines (SVM) classifier with Radial Basis Function (RBF) kernel achieved an accuracy of 72%, and a kNN classifier with over 95% accuracy when a low bit-rate was smaller than 400KB representing videos with low quality and size.

2.6 Machine Learning

ML is the art and science of programming computers so that they can learn from data [20]. Supervised ML (SML) uses algorithms to learn how to map a function from an input variable x to the output y. By approximating a function f as accurately as possible, when a new input x is given, the output y can be predicted such that $y = f(x)$. The components $[x_1, x_2, ..., x_n]$ of x are known as *features*. The two main tasks used by Supervised Machine Learning (SML) are classification and regression [20].

Following previous work from Panchenko et al. [15] and Wang et al. [25], this research extracted features such as packet time, packet length, and packet direction to predict labels such as the name or length of a YouTube video. These features were collected in a dataset to be able to train SML models. In addition, the VF attack model was evaluated in a closed-world setting, which assumes the user is viewing predetermined videos. One approach was to perform an 80/20 random split of the data into a training and test set respectively, in which the test set is used to test the model that was trained on the training set [20].

2.6.1 Classification Models

This research used the name of a video as the *label* for a *classification* task, and learn a function f that may *categorize* the output y from observing a new input x. The accuracy, precision, and recall will help analyze the performance of the classification models. As seen in Table 2.1, the four possible outcomes are represented in the confusion matrix. The model predicts *True Positives* and *True Negatives* when the prediction is the same as the label. The model predicts *False Positives* and *False Negatives* when the prediction is positive compared to a negative label, and when the prediction is negative compared to a positive label, respectively [20].

	Predicted Class = Yes	Predicted Class = No
Actual Class = Yes	True Positive	False Negative
Actual Class = No	False Positive	True Negative

Table 2.1. Confusion matrix [20].

Accuracy is the "ratio of correctly predicted observations to the total number of observations" [26], seen in Equation 2.1.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

Precision is the "ratio of observations correctly predicted divided by the total number of positive observations" [26], seen in Equation 2.2.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall is the "ratio of the positive observations correctly predicted divided by the total observations" [26], seen in Equation 2.3.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

K-Nearest Neighbor (kNN) In [20], Geron describes kNN as one of the most simple ML classification algorithms. In the research we use a given N training sample where the algorithm iterates through the training set and measures the *Euclidean Distance* from a new sample to all the other data points. By selecting the "k" nearest neighbors, and allowing each neighbor to have an equal vote, the new sample's class is determined by the majority.

In addition we use the weighted kNN where each *neighbors'* vote carries a weight that is inversely proportional to the distance from the new sample i.e. votes from closer neighbors count more than votes from neighbors that are farther away. Being a hyperparameter of the model, "k" may be tuned to achieve the best model fit.

Random Forest Classifier (RFC) The Random Forest algorithm works as a large collection of decorrelated decision trees [20]. Breiman's paper describes the selection of features [27]. We use a similar concept to create a training matrix composed of a large number of subsamples in which features such as direction and time packets are randomly chosen for each random decision tree that is created. Individual trees are limited to a depth of no more than two nodes, causing each tree to be a relatively *weak learner*. However, bagging the set of trees together and taking the average prediction produces a model that generalizes better over new test samples and improves overall prediction accuracy.

Support Vector Machines (SVM) As Wang describes [28], SVMs are a "distance-based metric function, which, given an instance, produces a value that describes which side of the decision boundary the instance should reside." Using Wang et. al's approach to classify websites, our research uses a similar (SVM) approach to classify video "traffic instances" where the function finds distance based similarities in the packet time or direction of our YouTube traffic in order to classify the instances.

Convolutional Neural Network (CNN) CNNs is a class of deep neural networks. As explained by Gall, "a neural network has an *input layer*, *hidden layers*, and an *output layer* where the *input layer* accepts inputs in different forms, while the *hidden layers* perform calculations on these inputs, and the *output layer* delivers the outcome of the calculations and extractions." [29] As websites like YouTube have adopted video streaming techniques such as DASH, research has shown that CNN layers can be used to generate representations of features from the unique burst patterns that a YouTube video contains [20], [22].

2.6.2 Regression Models

In another experiment, the packet length of a video will be used as the *label* for a *regression* task, and an estimation function f will be trained to regress a *numerical* output y from observing a new input x. Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 metrics explained below were used to analyze the performance of the models [20].

MAE "represents the average of the absolute difference between the actual and predicted values. It measures the average of the residuals in the dataset." [30] As seen in Equation 2.4,

$$\mathrm{MAE} = \frac{1}{n}\sum_{1}^{n} |h(x) - y| \tag{2.4}$$

the residuals are the vertical distance between the predicted value $h(x)$, and the label y. The absolute difference of the residuals is averaged over the dataset to give MAE.

MSE "represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals" [30], seen in Equation 2.5.

$$\mathrm{MSE} = \frac{1}{n}\sum_{1}^{n}(h(x) - y)^2 \tag{2.5}$$

RMSE "is the square root of mean squared error. It measures the standard deviation of residuals" [30], seen in Equation 2.6.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{1}^{n}(h(x) - y)^2} \tag{2.6}$$

R^2 or coefficient of determination "represents the proportion of the variance in the dependent variable which is explained by the linear regression model. It is a scale-free score" [30], seen in Equation 2.7. Values from 0 to 1 are interpreted as percentages. The higher the value is, the better the model is.

$$R^2 = 1 - \frac{\sum_1^n (y^n - y')^2}{\sum_1^n (y - y')^2} \tag{2.7}$$

Gradient Boost Regressor (GBR) GBR is an ensemble learning technique in which a combination of several ML algorithms [31] known as weak learners [20], together are trained with the effort to reach higher predictions compared to what a single ML model could achieve. This approach uses a boosting technique where the predictors are trained in a sequential order, each learning the error rate produced from its predecessor [20].

Random Forest Regression (RFR) Similar to GBR, RFR is an ensemble of Decision Trees trained by a bagging technique [20]. In bagging, ML algorithms are trained independently on sub-samples of the training set, creating a robust approach vice using a single ML algorithm. By averaging the individual model's output, the overall predictions of the ensemble is improved [32]. This technique improves variance for the ensemble as whole, at the expense of increased bias for each individual model within the ensemble; where $bias$ is a generalization of errors due to wrong assumptions and the $variance$ is a model that is perceptive to small variations in a training set [20].

K-Neighbors Regressor (kNR) kNR is another powerful regression model that may be used to predict a continuous value such as the video length of a YouTube video. kNR finds the nearest neighbors in a non-linear data distribution that matches the query point. It approximates the association between independent features by averaging the closest observation in the same neighbor. The prediction is the average of the observed values [33]. It is important to mention that the selection of the k value affects the outcome by overfitting the data, which means that the variance of the error estimate could be high. Therefore, choosing k requires testing on a range of k values, and selecting the value that generalizes the best over test samples without overfitting the training data.

CHAPTER 3: Methodology

This chapter discusses the data collection and preparation process required to conduct an evaluation of machine learning and its application for predicting YouTube videos. Additionally, this chapter provides initial data representations and explores assumptions for using the WF attack models and other ML models to achieve the VF attack.

3.1 Data collection

Using a similar web-crawler concept that Wang and Juarez applied to the WF attack model [23], [28], we collected network traces from a home Wi-Fi network environment to simulate the VF attack model. The steps illustrated in Figure 3.1 are the visual representation of the Virtual Machine (VM) environment. The process to collect the data was divided into five separate phases.

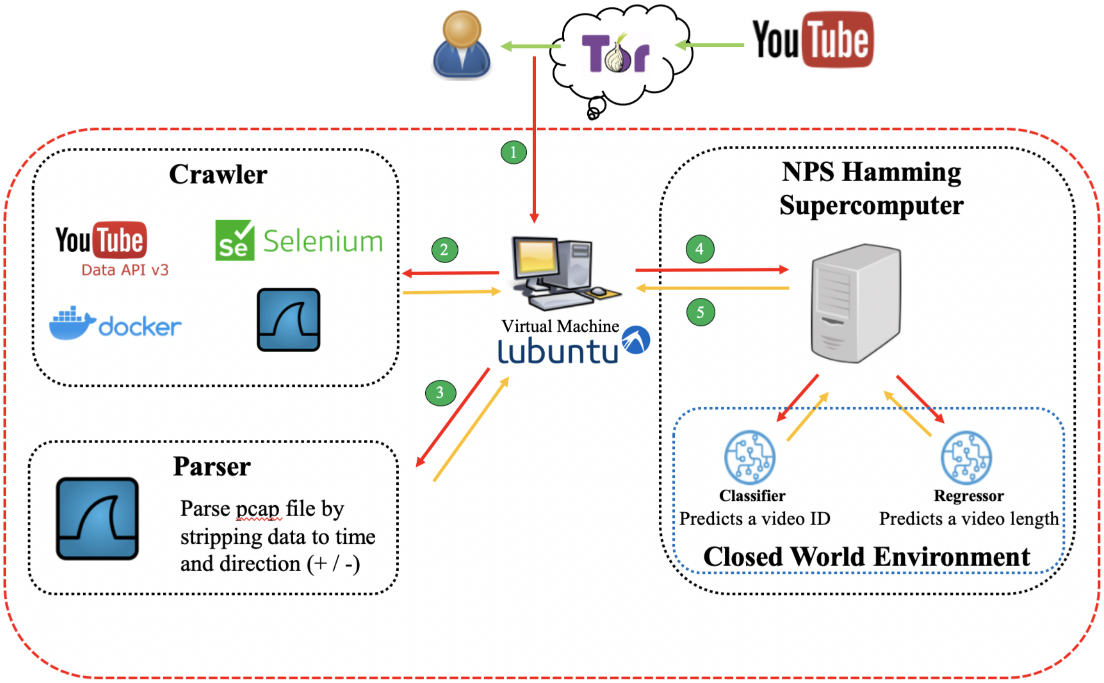


Figure 3.1. Virtual Machine environment.

Phase 1 The collection and selection of the metadata was done in a VM environment. Lubuntu[1] version 20.04 VM operating system allowed the use of a lightweight sterile environment with 350 Megabytes of RAM and low hardware requirements where YouTube videos were passively downloaded from a Tor network.

Phase 2 The crawler [34] uses Selenium[2] to automate the Tor Browser Bundle (TBB) version 8.0.2 and the YouTube Data Application Programming Interface (API)[3] to stop the video capture when done playing. The Docker[4] software helps run the crawler inside an encapsulated package called `tbcrawl` container. In addition, the crawler uses *tcpdump*, a program designed to capture network packets for analysis. It is important to mention that a YouTube video will always play an advertisement before the user can watch the selected video, the crawler is set to skip this option to recreate a user interaction with YouTube.

The configuration of a Tor circuit is set to close in 600000 (seconds), our approach follows Wang and Goldberg [28] where the circuit is set to close in 10 minutes. By setting the `UseEntryGuard` to 0, it prevents a user from using a fixed set of guard nodes, forcing the guard node to be selected in a random fashion. Since a user is not expected to make changes to these types of configurations, the crawler model proceeds to emulate a real collection of data. Through Wireshark[5], a network protocol analyzer, Packet Capture (PCAP) files are saved for future use in a parser code, which is explained in Phase 3. Lastly, the crawler takes screenshots every so often to let an adversary visually verify that data from a YouTube video is being collected.

Phase 3 The next phase uses a parser code to identify and separate the data as needed. By using the Scapy library to parse the captures and TShark to filter the traces by IP/MAC address, the parser code processes IP-level capture PCAP files into a sequence of tuples containing packet time and direction. The parser code recursively searches through directories to find PCAP files. For each PCAP file, it parses through and fetches timestamp, packet size, and direction. Applying Wang et al.'s [17] approach, we converted the raw packet direction values to +1 for outgoing and -1 for incoming as observed from the clients perspective.

[1] https://lubuntu.net/

[2] http://docs.seleniumhq.org/

[3] https://developers.google.com/youtube/v3/

[4] https://www.docker.com/

[5] https://www.wireshark.org

We also kept a copy of the raw times and directions for additional analysis as discussed in Section 4.3.

Phase 4 Due to the high computation cost of collecting and processing metadata from a home WiFi network, a parallel computing cluster was used to perform data exploration. The Hamming supercomputer used by Naval Postgraduate School (NPS) provided enough computer resources to parse the raw data into the required formats for the next research steps. The Hamming resources used for this work included one node with 250GB of memory and 64 CPUs which allowed multiple SML models to run simultaneously.

Using Wang et al. [28] methodology, the "data was collected batch-by-batch, with each batch corresponding to one circuit (one client)"; in our exploration, each batch took around seven hours to download. By crawling and downloading the nine YouTube videos into a batch, we collected each video 10 times. This resulted in a collection of 138 batches in the course of seven months. Following Wang's steps, "if the size of the video traffic instance was less than 20% of the median size for that video, it was removed." This is an anomaly we experienced same as Wang where he describes them as "failed instances, which may be a failed connection to the server or a server-originated message that denied access to the client." We also removed samples that downloaded less than 100 packets.

Phase 5 The last phase is run in a closed world environment where the video ID and video length are used as labels in a training and test set. The dataset was created and stored in a Pandas Dataframe with rows corresponding to each video recording, and the columns corresponding to video identification, video length, last packet time, individual packet times $\{t_0, t_1, t_2, ...t_n\}$, and individual packet directions $\{d_0, d_1, d_2, ...d_n\}$. To deal with videos of different lengths, we added padding in the form of zeros to the end of all samples to ensure that the feature inputs should be the same size. This resulted in a total of 86,754 features for all samples.

We used Equation 3.1 [20], which is described below, to scale the x values to help improve the ML model.

$$x' = \frac{x - \overline{x}}{\sigma} \tag{3.1}$$

First, we calculated the mean and standard deviation for every column. Then, the distribution mean was shifted to zero by taking the column value and subtracting the mean. The distribution was then divided by the standard deviation to bring the variance around one for all the columns. This scaling procedure allows the data to be in the same scale rather than having high and low values in the columns. As for the *y* values, we used the video identification for the classifier ML algorithms and video length for the regressor ML algorithms.

3.2 Dataset

For this research, we used the YouTube website, a popular video-sharing platform viewed by millions of people. We collected nine popular movie trailers with different *video lengths* in the range of two to seven minutes to create the dataset. These short videos were selected with the assumption that an extremist group may post short propaganda videos that could be seen by a regular user who uses YouTube as their search mechanism to search short videos such as movie trailers. We attempted to achieve similar results as Herrmann et al. [8] WF traffic analysis attack model, where the labels are webpages and the features are traffic traces. In our approach, we used the video ID and video length of the YouTube videos as the labels, and traffic traces as the features for a VF attack.

3.3 Threat Model

Following Ever, Juarez, Pachenko et al. [5], [11], [15] assumptions for a WF attack model, we explored the same assumptions for a VF attack model as discussed below.

We assume that the DoD or other agency impersonate as an adversary that has the capability to compromise an ISP or is able to access the WiFi of the client to passively observe traffic. This approach recreates how an adversary can intercept the traffic trace without modifying or decrypting the transmission to predict information about the YouTube video the client is watching over a Tor network.

In this environment, we assume that an adversary compiled a series of popular movie trailers watched by the client in recent months. This research considers Juarez's [23] criticism for a WF attack for not being a reflection of a real-world scenario. In a WF attack model, an adversary is not able to collect data from all the popular websites visited by a client. In our case, an adversary with high resources will never be able to collect traffic traces for all the videos YouTube has in its inventory. Instead, the assumption is that an adversary could use a small training set as an initial approach to visualize the dataset and find possible anomalies.

By taking into consideration the relationship between download times and the actual play time of each video, as well as between download times and the possible network latency that a regular client may encounter, we attempt to predict the play time of a video by observing a set of downloaded traffic traces and inspecting the direction of incoming and outgoing packets.

Based on the amount of data collected, and the use of a specific version of the TBB (ruling out software updates), we assume that the VF attack model will be as effective as the WF attack model as shown in previous research [4], [15].

However, some concerns other researchers [35] have noted are:

- An adversary may not know if the client is playing multiple videos simultaneously.
- Downloading other files when the adversary is collecting data could create noise traffic. Although this research does not work on these concerns, it is mentioned as a concern that other researchers have worked on for the WF attack models [19], [36], [37].

3.4 Attack Model

The intention of the adversary is to challenge the defense mechanism that Tor offers to a client. As seen in phase 1 of Figure 3.1, by impersonating an eavesdropper who collects several batches of YouTube videos into a dataset, we are able to analyze the traffic between a client and the guard node of a Tor network.

After collection, the adversary extracts features from a dataset of nine YouTube videos with length averaging between two and seven minutes. Some features that could help predict a YouTube video ID or length are: 1) packet transmission time, and 2) incoming or outgoing direction. Since this approach has been successful on WF attack models, an adversary could potentially use the same features for a VF attack. By using these features in a closed world scenario, the adversary can monitor YouTube videos that a user possibly watches. The exploration assumes a real scenario where an adversary knows the videos that a user is watching.

CHAPTER 4:
Test Design and Implementation

This chapter describes in-depth the five phases described in Chapter 3. It implements *Scikit-learn* [38] for the classification and regression algorithms, and *Tensorflow* [39] for the CNN algorithms. We explore the learning pipeline with the selected features and show how the system was configured to produce ideal results.

4.1 Data Generation

By populating the dataset from the 2-tuples, packet times and directions, we experimented with the performance of each of the algorithms. Using both packet times and directions as features, and solely packet times and solely packet directions as features, we were able to find strengths and weaknesses for each of the algorithms.

Before evaluating the algorithms in Section 4.5 and 4.6, we *scaled* and *split* the dataset into a train and a test set. We used an 80-20 split where 80% of the data was used for training, and 20% was used for testing [20]. In addition, the training set was used to *fit* the model, where the data in the model is generalized as close as possible to a similar label. The test set was set aside and used later to predict the unseen data. Additionally, outliers were removed to improve the performance of the ML algorithms. Videos showing *last packet time* capture with unusual sizes that were not close to the actual *video length* were removed from the dataset. The action resulted in the removal of 5920 video captures leaving 5605 captures in the dataset for exploration.

4.2 Initial Exploration

Initial visual exploration of the dataset shown in Figure 4.1, with a *Heatscatter* plot, reveals a slight positive correlation between last packet time and video length. Using the dataset with all nine videos, we explored the correlation between the video length represented as *length* on the y-axis, and the time the last packet was received for the video download, represented as the *last packet time* on the x-axis. The last packet time should be a good indicator of video length because it marks the time at which the download is complete.

Moreover, last packet time will always vary due to download speed, network latency, and congestion. Figure 4.1 shows from the start of each video an increase in last packet time as video length increases indicating a positive correlation.

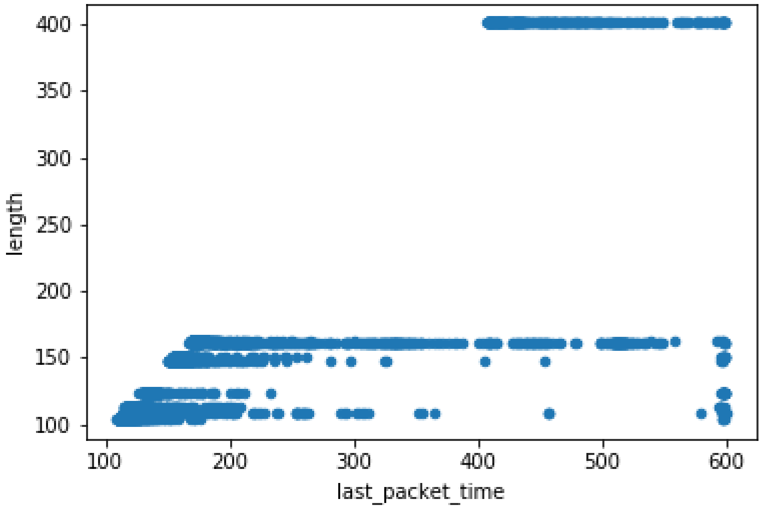


Figure 4.1. Initial visualization of nine video lengths and their last packet time captured. In a positive linear correlation, we expect to see the *y* values increase as the *x* values increases [20].

Continuing the exploration, Table 4.1 shows more precisely the numerical representation of existing correlation values ranging from -1 to 1 [20]. The table shows a fairly strong positive correlation of 0.73 between last packet time and video length. Additionally, we see a slight positive correlation between last packet time and video ID of 0.39. These results suggest that last packet time should be a strong predictor for our ML algorithms. Moreover, the table indicates that some features are correlated. For example, there is a correlation of 0.52 between t100 and t500. This suggests that the selected ML approach should not assume zero correlation among features. For example, since Logistic Regression assumes zero correlation among features, it may not be the best approach for this dataset [20].

	length	**last pkt time**	**t100**	**t500**	**d100**	**d500**	**video ID**
length	1.00	0.73	-0.02	-0.01	-0.00	-0.01	0.39
last pkt time	0.73	1.00	0.01	0.06	0.01	-0.02	0.41
t100	-0.02	0.01	1.00	0.52	0.04	0.03	-0.04
t500	-0.01	0.06	0.52	1.00	0.03	0.02	-0.01
d100	-0.02	0.01	0.04	0.03	1.00	0.01	0.05
d500	-0.01	-0.02	0.03	0.02	0.00	1.00	-0.01
video ID	0.39	0.41	-0.04	-0.00	0.05	-0.01	1.00

Table 4.1. Correlation matrix.

Another approach for understanding correlation was by visualizing the correlation matrix using the *Seaborn Heatmap* shown in Figure 4.2. The color bar chart displays dark colors indicating low correlation, and light colors indicating high correlation. These values indicate the same findings from Table 4.1 which are: 1) last packet time has a positive correlation with video length and video ID, and 2) some features are correlated in the dataset.

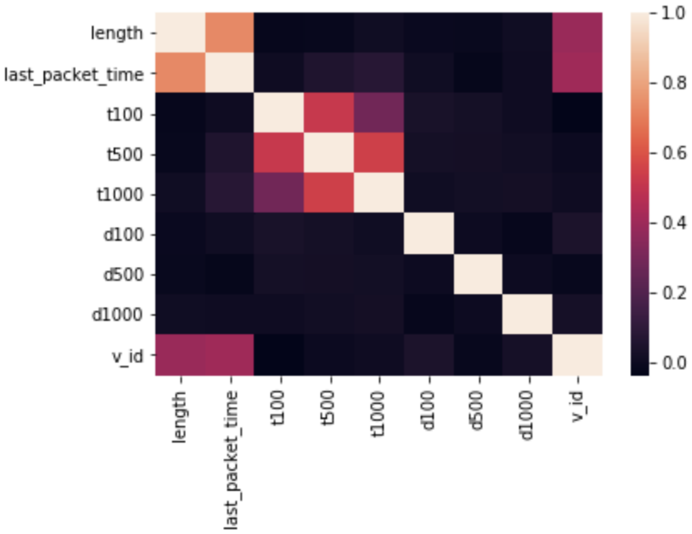
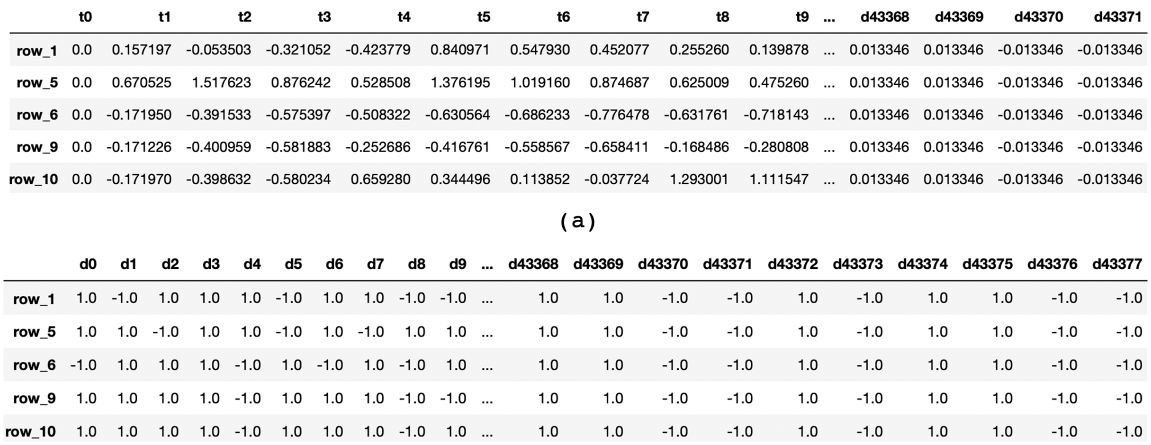


Figure 4.2. Correlation matrix for different features and labels of the dataset.

4.3 Data Format

We used two individual approaches to structure the dataset in preparation for the classifier and regressor ML algorithms. Following Wang et al. [25], we used packet direction and packet time as independent *x* values, and combined the *packet time* with the *last packet time* feature to explore which algorithm will provide higher performance. Figure 4.3 (a) shows a representation after the data was scaled. Using the same dataset, we followed Bhat et al. [35] and removed times from the dataset, and only kept the sign of the packet direction. This approach removes the magnitude and only considers direction. The transformation is shown in Figure 4.3 (b) where incoming and outgoing packets are represented by +1 and -1 respectively. We found that method (a) produced the best results for all ML algorithms other than CNNs, and method (b) produced the best results for CNNs.



	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	...	d43368	d43369	d43370	d43371
row_1	0.0	0.157197	-0.053503	-0.321052	-0.423779	0.840971	0.547930	0.452077	0.255260	0.139878	...	0.013346	0.013346	-0.013346	-0.013346
row_5	0.0	0.670525	1.517623	0.876242	0.528508	1.376195	1.019160	0.874687	0.625009	0.475260	...	0.013346	0.013346	-0.013346	-0.013346
row_6	0.0	-0.171950	-0.391533	-0.575397	-0.508322	-0.630564	-0.686233	-0.776478	-0.631761	-0.718143	...	0.013346	0.013346	-0.013346	-0.013346
row_9	0.0	-0.171226	-0.400959	-0.581883	-0.252686	-0.416761	-0.558567	-0.658411	-0.168486	-0.280808	...	0.013346	0.013346	-0.013346	-0.013346
row_10	0.0	-0.171970	-0.398632	-0.580234	0.659280	0.344496	0.113852	-0.037724	1.293001	1.111547	...	0.013346	0.013346	-0.013346	-0.013346

(a)

	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	...	d43368	d43369	d43370	d43371	d43372	d43373	d43374	d43375	d43376	d43377
row_1	1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0	...	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0
row_5	1.0	1.0	-1.0	1.0	1.0	-1.0	1.0	-1.0	1.0	1.0	...	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0
row_6	-1.0	1.0	1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	1.0	...	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0
row_9	1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0	-1.0	-1.0	...	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0
row_10	1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	...	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	-1.0

(b)

Figure 4.3. a) Scaled dataset representing packet times and directions. b) Dataset with packet directions only, i.e., +1 representing incoming, and -1 representing outgoing packets.

4.4 Voting Classifier

The voting classifier uses two ensemble learning techniques known as *hard* voting classification and *soft* voting classification. As the name implies, the *hard* voting classifier votes for the model with the maximum number of outputs that predict the label. In contrast, the *soft* voting classifier makes a prediction by calculating the average of all the probabilities when comparing the models [20].

By training the Random Forest Classifier (RFC), the Support Vector Machine Classifier (SVM), and the Logistic Regression Classifier (LRC) with an Ensemble Voting Classifier using the *packet times* as the *x* value and *video identification* as the *y* value, we see in Figure 4.4 that the LRC had the lowest accuracy. This may be due to the features of the model having non zero correlation. Because of the low LRC accuracy, the Voting Classifier Ensemble did not reach the highest accuracy that it normally achieves. Instead, our initial approach shows that the RFC had the highest accuracy of 74% [40]. These results suggest that an adversary may predict the video a user watches over Tor with 74% accuracy in a closed-world attack model.

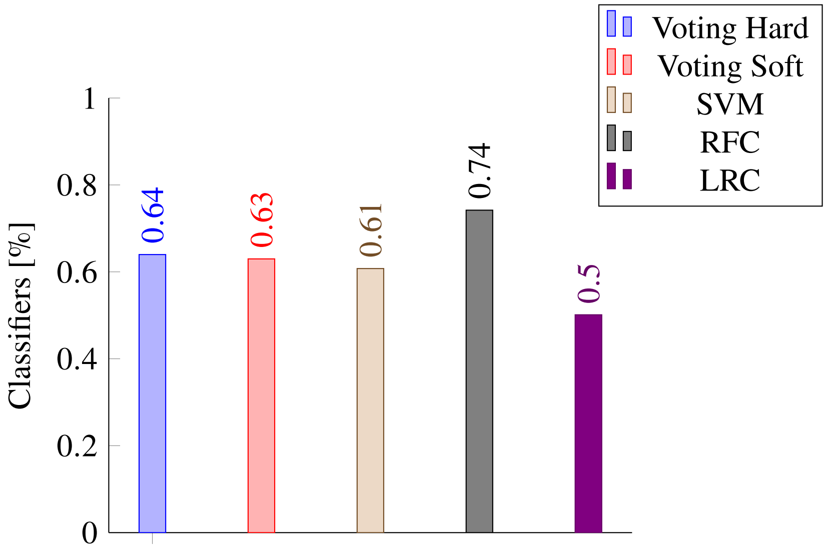


Figure 4.4. Accuracy of four classifiers on predicting YouTube video ID using `packet times` as features.

4.5 Machine Learning Classifiers

We now explore further into predicting the YouTube video ID using additional ML algorithms with various feature sets. Moreover, we explore datasets composed of batches of *video ID*, *packet direction*, *packet time* and, *last packet time*.

4.5.1 Support Vector Machine (SVM)

The SVM model is trained in a SML [20], [41] environment. The exploration uses the `RBF` kernel function where it uses a parameter γ that scales the features to be greater than zero. The kernel assesses how much influence each observation has in the training set to classify the observation in a test set. By using two points, x and x', `RBF` squares the difference between them, which is represented as the amount of influence one observation has on the other. By using γ, the squared distances are scaled, thus scaling the influence. This method ensures that observations far from each other are not selected as the influence [41]. Another approach to help tune the parameters is the use of the `linear` kernel function where the data is separated by a line between two x points, i.e., x and x'. This approach allows testing the data much faster than the `RBF` kernel as it uses only the c parameter to optimize the observation.

By using γ and c parameters for the `RBF` kernel, the model is trained with various combinations, such as c = {1, 10, 100, 1000} and γ = {0.001, 0.0001} respectively. In addition, we apply `GridSearchCV` where the number of estimators were fit to the model training set. For the `linear` kernel we follow the same steps using c = {1, 10, 100, 1000}. The result with the highest accuracy, using an x shape of (5229, 43378) and y shape of (5229, 1) was 72% when using the `RBF` kernel function with *packet time* as the features. The accuracy dropped to 43.9% when only *packet directions* where used as features, and to 50.50% when only *last packet time* was used.

Figure 4.5, is a visual representation of the `RBF` kernel function where the parameters c=10^3 and γ=10^{-3} had the highest accuracy compared to other c and γ parameter values.

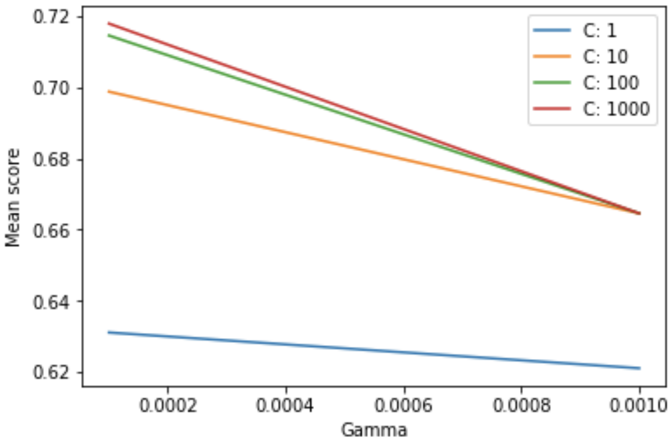


Figure 4.5. Visualization of the best C and γ parameters for an SVM algorithm.

4.5.2 Random Forest Classifier(RFC)

Using the Decision Tree Classifier (DTC) as the initial approach, we analyzed the accuracy for predicting video ID for the feature sets: *packet direction* and *packet time*. In Table 4.2, the models trained with only *packet direction* features, and only *packet time* features are labeled as models (1) and (2) respectively. As seen in Table 4.2, we observed the highest accuracy of 63.7% when using *packet time* features.

Initial Shape	**Train** **Test shape**	(1) **`direction`** (2) **`time`**	**Accuracy**
(5616, 86759)	Train (4492, 86757) Test (1124, 86757)	(1) (1) (2) (2)	0.322 0.567 **0.637**

Table 4.2. Decision Tree analysis of different traffic features. The highest accuracy is achieved when the model is trained with `packet time` features.

We performed the same experiments using a Random Forests Classifier, that is, a bag of Decision Trees where the growth of each tree is limited and controlled by the *max depth* parameter [20]. We applied the `GridSearchCV` function and continued using *packet time* and *last pkt time* as the feature set. We trained on a range of parameters such as `n estimators` = {50, 100, 300, 500, 800, 1200}, `max depth` = {2, 3, 5, 8, 10, 15, 25, 30, 50, 100}, `min samples split` = {2, 5, 10, 15, 100}, and `min samples leaf` = {1, 2, 5, 10} with a dataset using an *x* shape of (5605, 43378) and a *y* shape of (5605, 1).

Next, we used the `feature importance` function that Random Forests provides. The model is first trained, then the most important features are scored from values ranging from 0 to 1. Values closer to 1 indicate features that contribute more to the classification. In our analysis, the model was improved by removing 30499 features with score of 0 from the total 43378 features. The model was trained on a range of parameters such as `n estimators` = {2, 50, 100, 300, 400, 500, 600}, and `max leaf nodes` = {2, 50, 100, 300, 400, 500, 600}. Max leaf nodes and n estimators were also limited to prevent possible *overfitting*. These steps led to the highest RFC accuracy score of 78%.

4.5.3 k-Nearest Neighbor Classifier(kNN)

For this classifier, we used the `GridSearchCV` function with *packet times*, and *packet direction* feature sets. The dataset shape that gave the highest accuracy had *x* shape of (5434, 43378), and *y* shape of (5434, 1).

New samples were classified by taking the shortest distance from the sample to the *k nearest neighbors*. We tuned the parameter *k* from the values *k* = {3, 5, 11, 19} and settled on the model that produced the highest accuracy. This step ensured the data did not have *outliers* due to low *k* values or the data was not *overfitting* for large *k* values. Another parameter that was tuned was the `weight` parameter used for weighted k-NN. This parameter allows neighbors' votes to count more if they are closer to the test point, and less if they are farther away. *Euclidean Distance* was used to find the distance between any two points in the dataset. `Manhattan Distance` was also an optional distance metric in which the distance of two points is calculated as the sum of the absolute differences of their *Cartesian* coordinates. To control the different parameters selected, we used the `auto` option to determine the best approach, and `K-D tree` to reduce the number of distance calculations from the training set [20], [42].

The use of the parameter grid allowed us to test all these different parameter selections. The parameters that allowed us to reach a high accuracy were `Manhattan Distance`, `weight` and the `auto` option. The outcome produced with the highest accuracy score came from the features *packet time* with a 70% of accuracy score.

4.5.4 Convolutional Neural Network (CNN)

This research replicates a DF attack model that Sirinam and Juarez et al. [5], [20] used for a WF attack. Tor uses lightweight defenses such as *W-T* and *WTF-PAD* as a countermeasure to alleviate low latency overhead. Their work evaluates a DL attack model against website traffic traces with *W-T* defense, *WTF-PAD* defense, and a defenseless Tor traffic called *Non-defended (NoDef)*.

Using Schuster et al.'s [22] proposal that CNNs may be used in an encrypted video stream due to the unique characteristics of their burst patterns, we proceed to use CNNs for a VF attack to predict YouTube videos over Tor. Applying previous steps to ensure the model is not overfitting, the dataset was `scaled` and `split` into a train, a validation, and a test set, with a ratio of 8:1:1 respectively. The packet direction was used for the *x* values, and the corresponding YouTube video IDs for the *y* values. For this model, we followed Wang et al.'s [17] approach in which the dataset was transformed to +1 for outgoing, and -1 for incoming packets as shown in Figure 4.3 (b).

The closed-world assumption was used for the attack model against all three Tor defenses. For the *NoDef* dataset, the model was composed of YouTube video traces with no Tor defenses against a VF attack. Considering that a classifier improves its learning by using more training *epochs*, we found that 60 training *epochs* achieved the highest accuracy levels. From an early stage, we noticed that the shape of the dataset impacted the accuracy percentage to predict the video ID. As seen in Figure 4.6, we notice that the classifier improve its learning to predict videos when the train sample increases from 1320 to 3810 samples, improving the accuracy percentage to 84%.

The exploration of the *WTF-PAD* Tor defense presented a positive outcome. We were able to reach an accuracy score of 84% with a train sample of 3577 samples. As for the *W-T* defense model, results improved as the train sample increase through the laps of our exploration. The final result had 83% for *W-T* and 92% for *W-T (Top-2)*, confirming that a neural network model is a strong classifier when comparing with previous models.

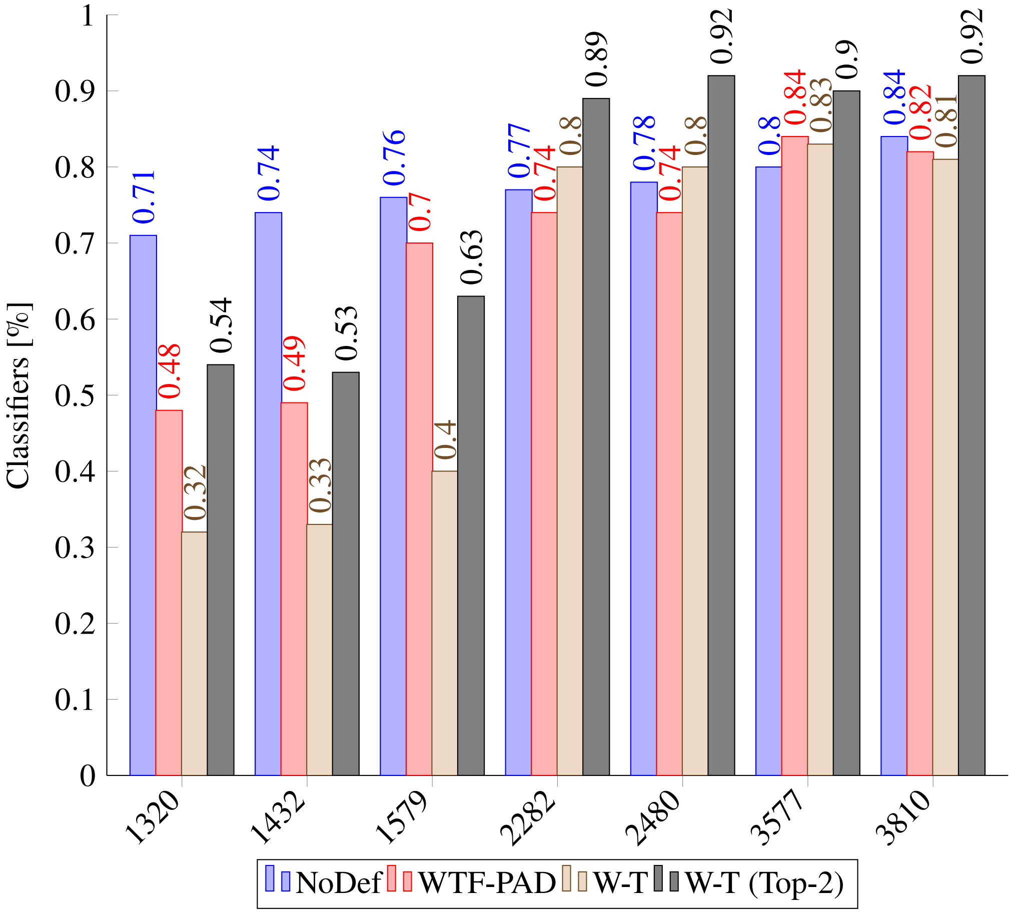


Figure 4.6. Performance of seven months on predicting YouTube video identification using `direction` as its feature.

4.6 Machine Learning Regressors

Predicting the video length could help aid classification tasks further down in the learning pipeline. Furthermore, if an adversary is capable of predicting video length, they may be able to improve classification accuracy in an open-world attack model by filtering out videos that are significantly longer or shorter than videos of interest.

Similar to the ML Classifiers, we used datasets composed of *packet direction* and *packet times*, which were split into a train and test set. The length of each of the YouTube videos was used as the continuous dependent variable. Further exploration of the Regressor models resulted in higher R^2 values when *packet times* were used for all three regressors rather than both packet times and directions.

4.6.1 Gradient Boosting Regression (GBR)

Using the ensemble method, as seen in Table 4.3, the MAE shows that GBR is able to predict a YouTube video length with an error of ±7.03 seconds.

GBR Mean Absolute Error	7.0326
GBR Mean Squared Error	288.3639
GBR Root Mean Squared Error	16.9813
GBR R^2	0.9396

Table 4.3. Performance model for Gradient Boosting Regression.

4.6.2 k-Neighbors Regressor (kNR)

For the kNR model, we used packet times as a feature and dropped the remaining columns, *packet direction*, and *last packet time*. The model used the nearest neighbor k=3 with `weights` and `Manhattan Distance` as the distance metric. With these settings, kNR was able to predict a video length with ±5.2861 mean absolute error.

kNR Mean Absolute Error	5.2861
kNR Mean Squared Error	293.0589
kNR Root Mean Squared Error	17.1190
kNR R^2	0.9238

Table 4.4. Performance model for k-Neighbors Regressor.

4.6.3 Random Forest Regression (RFR)

As shown in Table 4.5, the MAE is ±6.3521 which shows that RFR could be a second choice after kNR when predicting the video length of a YouTube video.

RFR Mean Absolute Error	6.3521
RFR Mean Squared Error	297.1422
RFR Root Mean Squared Error	17.2378
RFR R^2	0.9318

Table 4.5. Performance model for Random Forest Regressor.

4.7 Result Summary

By evaluating the performance of different ML classifiers and regressors with the different input traces *packet times* and *packet directions*, we were able to find from early on that regressors provide strong outcomes in predicting a video length. On the contrary, since the traffic pattern of a video streaming is dynamic due to DASH, our exploration shows that some of the classifiers such as kNN and SVM did not provide noticeable improvement in predicting video identification like the RFC or the neural network models.

An approach that an adversary might take is to use the kNR regressor model to help improve the weaker classification models. Since the MAE result for the kNR had the lowest time to predict the length of a YouTube video, an adversary could use it to create a new dataset in which videos are classified based on their length, eliminating videos with a specific duration that a user does not watch. However, the results observed over the last seven months show that increasing the number of video captures could be a factor in improving the training on some of the ML models in a closed world environment, which is important to consider for an adversary attempting to improve the performance of the learning *pipelining*.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

This research explored the performance of VF attacks using different classifier and regressor models in a closed-world scenario. In addition, by replicating the DF attack model that Sirinam and Juarez et al. [5], [20] used for their WF attack, we can confidently say that the information of a video can be predicted. The DoD can use ML techniques to be one step ahead of terrorists by identifying YouTube videos that are streamed from the dark web.

5.1 Contributions and Future Research

Although the results do not represent the total number of videos YouTube has available, we are confident that our results show that a VF model is effective in predicting the different labels from a small number of YouTube videos. During the period of the seven months that this research took place, we noticed that the three regressor models were effective in predicting the length of a video. Figure 5.1 shows that the Gradient Boosting Regression, the Random Forest Regression and the k-Neighbors Regressor models were successful in a closed-world setting.

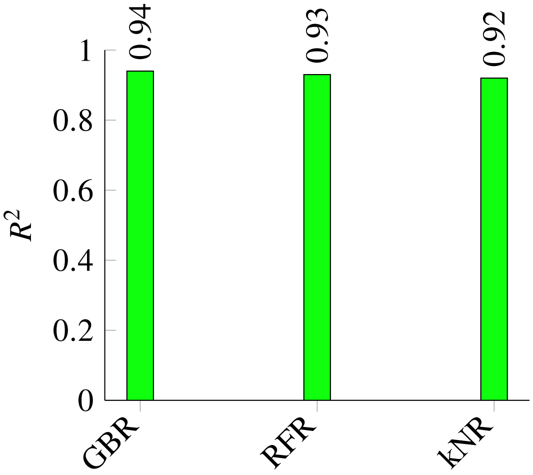


Figure 5.1. R^2 score of different machine learning regressors that were used to predict the length of YouTube videos.

Figure 5.2 shows that less complex models such as RFC, SVM, or kNN were not as successful as the deep learning models. Nevertheless, when comparing these models, we propose that a way to improve these conventional models could be by collecting more videos. Since we noticed that each of the models improved their accuracy score when the shape of the model increased, we believe that this approach could help improve the outcome of some of models.

One idea for future work is to test the attacks presented in this research using an open-world attack model. In these experiments, the user will be allowed to download videos that the adversary did not include in their training set. The adversary must include some form of an open-world class in their model to be successful. This experiment would require more collection of data. Perhaps two or three more video genres can be collected and used to train and test a classifier in an open-world setting.

Additionally, in this work we trained ML models to predict the video ID or regress the video length. One idea would be to study how well ML models may perform at predicting the video genre. If multiple genres of videos are collected, for example military videos, news stories, etc., classifiers can be trained and tested for their accuracy to predict the video genre to a specific viewer.

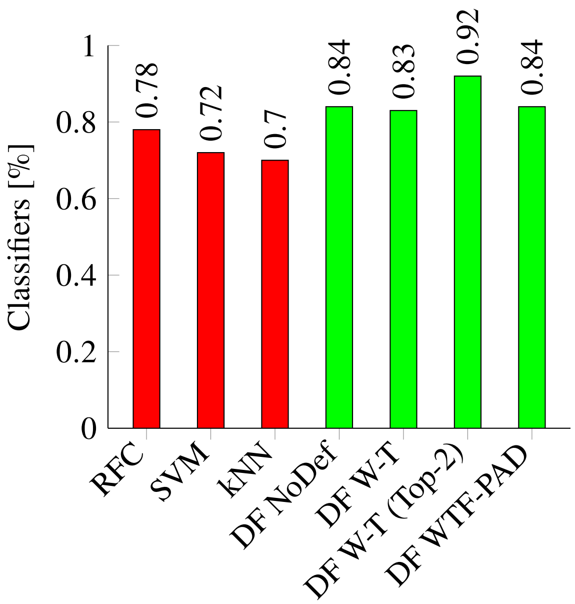


Figure 5.2. Accuracy scores of different machine learning classifiers for predicting YouTube videos over Tor.

5.2 Conclusion

This research was able to prove that passive adversaries can use traffic patterns to train SML models to predict a *video ID* or *video length* from a small set of possibilities. By creating a small dataset composed of nine YouTube videos that were passively downloaded from a Tor network, we were able to unmask video information from a user. Nevertheless, we propose that future research increases the size of the dataset. This is an approach other research has done for WF attack models [5], [8], [16], [20] and we are confident that this same methodology could be successful for a VF attack model. If the dataset is increased to a large number of videos, such as 100 videos, it could help improve the results in a closed-world scenario. In addition, an adversary could continue improving the parameters during the training cycle to deploy these models to an open-world scenario where the adversary is not aware of the videos a user is watching.

Overall, similar WF attack models, we found that our VF attack models are successful in finding information about a YouTube video downloaded over Tor. We show that the deep neural network models reach accuracy results in the 90% range which are similar to the result Sirinam and Juarez et al. [5], [20] had for their WF attack models. This tells us that a Tor network is susceptible to the VF attack when using an ML algorithm with enough features to predict a *video identification* or the *video length*. Either of these approaches could provide insightful information for an organization like the DoD when posing as an adversary to find specific videos posted by extremist groups on the Tor network.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] N. V. Denic, “Government activities to detect, deter and disrupt threats enumerating from the dark web,” US Army Command and General Staff College Fort Leavenworth United States, Tech. Rep., 2017.

[2] A. Gupta, S. B. Maynard, and A. Ahmad, “The dark web phenomenon: A review and research agenda,” *arXiv preprint arXiv:2104.07138*, 2021.

[3] X. Zhang and K. Chow, “A framework for dark web threat intelligence analysis,” in *Cyber Warfare and Terrorism: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020, pp. 266–276.

[4] M. S. Rahman, N. Matthews, and M. Wright, “Poster: Video fingerprinting in tor,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2629–2631.

[5] P. Sirinam, M. Imani, M. Juarez, and M. Wright, “Deep fingerprinting: Undermining website fingerprinting defenses with deep learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[6] H. Frystyk. (1994, July). The Internet Protocol Stack. [Online]. Available: https://www.w3.org/People/Frystyk/thesis/TcpIp.html/. Accessed Mar 2, 2021.

[7] X. Du, M. Shayman, and M. Rozenblit, “Implementation and performance analysis of snmp on a tls/tcp base,” in *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium (Cat. No. 01EX470)*. IEEE, 2001, pp. 453–466.

[8] D. Herrmann, R. Wendolsky, and H. Federrath, “Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier,” in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009, pp. 31–42.

[9] P. Syverson, R. Dingledine, and N. Mathewson, “Tor: The secondgeneration onion router,” in *Usenix Security*, 2004, pp. 303–320.

[10] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, “Towards an analysis of onion routing security,” in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 96–114.

[11] H. Evers, S. Kula, v. d. L. den Toom, and Pouwelse, "Thirteen years of tor attacks," Nov 2019. Available: https://github.com/Attacks-on-Tor/Attacks-on-Tor.git

[12] A. Barton and M. Wright, "Denasa: Destination-naive as-awareness in anonymous communications," *Proceedings on Privacy Enhancing Technologies PETS'16*, 2016.

[13] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 2005, pp. 183–195.

[14] G. Danezis, *Traffic Analysis of the HTTP Protocol over TLS*. Cambridge, United Kingdom: Citeseer, 2009.

[15] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 103–114.

[16] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.

[17] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 143–157.

[18] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, vol. 24, no. 11, pp. 1–10, 2015.

[19] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[20] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

[21] G. Fei and B. Liu, "Breaking the closed world assumption in text classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 506–514.

[22] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1357–1374.

[23] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 263–274.

[24] R. Dubin, A. Dvir, O. Hadar, and O. Pele, “I know what you saw last minute-the chrome browser case,” *Black Hat Europe*, 2016.

[25] W. Q. Wang and H. Shao, “High altitude platform multichannel SAR for wide-area and staring imaging,” *Aerosp. and Electron. Syst.*, vol. 29, no. 25, pp. 12–17, Mar. 2014.

[26] J. Dib, K. Sirlantzis, and G. Howells, “A review on negative road anomaly detection methods,” *IEEE Access*, vol. 8, pp. 57 298–57 316, 2020.

[27] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] T. Wang and I. Goldberg, “Improved website fingerprinting on Tor,” in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 201–212.

[29] R. Gall. (2018). What is a convolutional neural network (CNN). [Online]. Available: https://hub.packtpub.com/what-is-a-convolutional-neural-network-cnn-video/. Accessed April 19, 2021.

[30] A. Chugh. (2020). MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? [Online]. Available: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e/. Accessed April 11, 2021.

[31] D. Cournapeau, “sklearn.ensemble.gradientboostingregressor.” Accessed Apr. 20, 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html/

[32] D. Cournapeau, “sklearn.ensemble.randomforestregressor.” Accessed Apr. 10, 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html/

[33] D. Cournapeau, “sklearn.neighbors.kneighborsregressor.” Accessed Feb. 21, 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html/

[34] N. Mathews, “Tor-browser-crawler-video.” Accessed Oct. 18, 2021 [Online]. Available: https://github.com/notem/tor-browser-crawler-video/

[35] S. Bhat, D. Lu, A. Kwon, and S. Devadas, “Var-cnn: A data-efficient website fingerprinting attack based on deep learning,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 292–310, 2019.

[36] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 332–346.

[37] J. Hayes and G. Danezis, “k-fingerprinting: A robust scalable website fingerprinting technique,” in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 1187–1203.

[38] D. Cournapeau, “sklearn.” Accessed Sep. 28, 2021 [Online]. Available: https://scikit-learn.org/stable/

[39] G. B. Team, “tensorfl.” Accessed Nov. 8, 2021 [Online]. Available: https://www.tensorflow.org/

[40] D. Cournapeau, “sklearn.” Accessed Oct. 15, 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html/

[41] D. Cournapeau, “1.4. support vector machines.” Accessed Sep. 10, 2021 [Online]. Available: https://scikit-learn.org/stable/

[42] D. Cournapeau, “sklearn.neighbors.kneighborsclassifier.” Accessed Mar. 12, 2021 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html/

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California