Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning

Payap Sirinam
Rochester Institute of Technology
Rochester, New York
payap.sirinam@mail.rit.edu

Mohsen Imani
University of Texas at Arlington
Arlington, Texas
mohsen.imani@mavs.uta.edu

Marc Juarez
imec-COSIC KU Leuven
Leuven, Belgium
marc.juarez@kuleuven.be

Matthew Wright
Rochester Institute of Technology
Rochester, New York
matthew.wright@rit.edu

ABSTRACT

Website fingerprinting enables a local eavesdropper to determine which websites a user is visiting over an encrypted connection. State-of-the-art website fingerprinting attacks have been shown to be effective even against Tor. Recently, lightweight website fingerprinting defenses for Tor have been proposed that substantially degrade existing attacks: WTF-PAD and Walkie-Talkie. In this work, we present Deep Fingerprinting (DF), a new website fingerprinting attack against Tor that leverages a type of deep learning called Convolutional Neural Networks (CNN) with a sophisticated architecture design, and we evaluate this attack against WTF-PAD and Walkie-Talkie. The DF attack attains over 98% accuracy on Tor traffic without defenses, better than all prior attacks, and it is also the only attack that is effective against WTF-PAD with over 90% accuracy. Walkie-Talkie remains effective, holding the attack to just 49.7% accuracy. In the more realistic open-world setting, our attack remains effective, with 0.99 precision and 0.94 recall on undefended traffic. Against traffic defended with WTF-PAD in this setting, the attack still can get 0.96 precision and 0.68 recall. These findings highlight the need for effective defenses that protect against this new attack and that could be deployed in Tor. .

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; • **Networks → Network privacy and anonymity**;

KEYWORDS

Tor; privacy; website fingerprinting; deep learning

ACM Reference Format:
Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3243734.3243768

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243768

1 INTRODUCTION

With more than two million daily users, Tor has emerged as the de facto tool to anonymously browse the Internet [2]. Tor is, however, known to be vulnerable to traffic analysis. In particular, *website fingerprinting (WF)* is a traffic analysis attack with the potential ability to break the privacy that Tor aims to provide. WF allows the attacker to identify web pages in an encrypted connection by analyzing patterns in network traffic. This allows a local and passive network adversary, such as a user's Internet service provider or someone sniffing the user's wireless connection, to identify the websites that the user has visited despite her use of Tor.

WF exploits the fact that differences in website content (e.g., different images, scripts, styles) can be inferred from network traffic, even if traffic has been encrypted. From a machine learning perspective, WF is a classification problem: the adversary trains a classifier on a set of sites, extracting network traffic features that are unique to each website. To deploy the attack, the adversary uses the classifier to match traces of a victim to one of those sites. The effectiveness of WF depends heavily on both the classifier algorithm and the set of features used. Previous WF attacks use a set of hand-crafted features to represent Tor traffic, achieving 90%+ accuracy against Tor using classifiers such as Support Vector Machine (SVM) [27], k-Nearest Neighbors (k-NN) [38], and random forests [14].

In response to these attacks, a number of defenses have been proposed. WF defenses add dummy packets into the traffic and add delays to real packets, aiming to hide features exploited by WF attacks such as traffic bursts and packet lengths. Notably, Tor Project developers have shown an interest in deploying *adaptive padding* as a possible defense [29, 30]. Based on this, Juarez et al. proposed WTF-PAD and showed that it effectively defends against WF attacks with reasonable overheads, such that it would be practical for deployment in Tor [20]. Recently, Wang and Goldberg proposed another effective and low-overhead defense called Walkie-Talkie (W-T) [41]. These proposals raise the question of whether attacks could be improved to undermine the effectiveness of the new defenses, a question we address in this work.

While the state-of-the-art attacks use classifiers that are popular in many applications, *deep learning* (DL) has shown to outperform traditional machine learning techniques in many domains, such as speech recognition, visual object recognition, and object detection [23]. Furthermore, DL does not require selecting and fine-tuning features by hand [31]. In this work, we thus explore whether we can leverage deep learning to improve classification results against defended Tor traffic. The key contributions of our work are as follows:

- We propose Deep Fingerprinting (DF), a new WF attack based on a Convolutional Neural Network (CNN) designed using cutting-edge DL methods. The attack uses a simple input format and does not require handcrafting features for classification. We describe how DF leverages advances from computer vision research for effective and robust classification performance.
- To study the attack in detail, we experiment in a closed-world setting using a new dataset that we collected with 95 sites and 1,000 traces per site. We find that our DF WF attack is more accurate against Tor than the state-of-the-art attacks with 98.3% accuracy. We also show results for how the number of training epochs and training dataset size affect the classification accuracy.
- We then show the effectiveness of the DF attack in the closed-world setting against Tor traffic defended with WTF-PAD and W-T. Against WTF-PAD, the attack reaches 90% accuracy, which is significantly better than all other attacks. Against W-T, the attack reaches 49.7% accuracy, which is better than all other attacks and nearly the theoretical maximum accuracy [41].
- To investigate in a more realistic setting, we use an open world with 20,000 unmonitored sites. On non-defended traffic, the attack achieves 0.99 precision and 0.94 recall. On traffic defended with WTF-PAD, the attack yields 0.95 precision and 0.70 recall. We also examine the possibilities for attacking weak implementations of W-T.
- Based on our experimental findings, we propose a number of new directions to explore in both attack and defense.

Overall, we find that the new DF WF attack undermines at least one defense that had been considered seriously for deployment in Tor [29, 30]. We have disclosed our findings to the Tor Project, and they have expressed their concerns about WTF-PAD, setting the stage for more exploration of the design of realistic defenses.

2 THREAT MODEL

Among its goals, Tor aims to protect users against local eavesdroppers from learning what sites the user is going to. WF attacks, however, use traffic analysis to undermine Tor's protections. Prior work has shown that, under certain conditions, a local and passive adversary can identify the pages visited by a Tor user by exploiting patterns in network traffic [8, 14, 16, 27, 28, 39, 40].

To deploy the attack, the adversary captures the sequence of packets, also known as a *traffic trace*, from each of a series of his own visits to a representative set of websites, including sites he is interested in detecting. From each trace, he then extracts features that are unique to each website. In the WF literature, we find a myriad of such features: packet size frequencies [16], total transmission time and volume in both directions [28], edit-distance score [8, 39], and the number of traffic bursts in each direction [28, 38], just to mention a few. As a result, the adversary obtains several *feature vectors* for each website that are used to train a supervised classifier that learns how to identify the site from its features. Finally, the adversary can collect new traffic traces from the user's connection to the Tor network, extract the features, and use the trained classifier to guess the website.

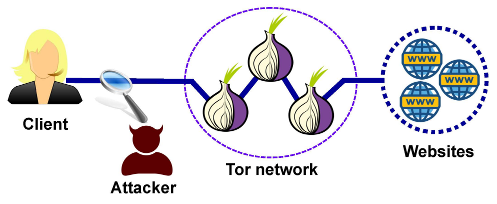


Figure 1: The WF threat model

In this work, we assume a network-level adversary that is: *local*, meaning that he has access only to the link between the user and the entry node to the Tor network, and *passive*, i.e., he can record network packets but not modify, delay, drop or decrypt them. Potential adversaries that might be in a position to deploy a WF attack include: eavesdroppers on the user's local network, local system administrators, Internet Service Providers (ISP), Autonomous Systems (AS) between the user and the entry node, and the operators of the entry node.

Figure 1 depicts the attack scenario: the client surfs the Web over the Tor anonymity system and the attacker intercepts the traffic between the client and the Tor network. We assume the adversary knows the client's identity and only aims at identifying the website. Note that the adversary can trivially obtain the client's IP address as long as he has access to the TLS connection between the user and the entry node. Beyond the entry node, Tor has stripped a layer of encryption and the IP of the client is no longer present in the headers of network packets.

Within this scenario, we draw on prior work to make several assumptions about the attacker goals and capabilities.

Closed- vs Open-world Scenario: A closed-word assumes the user can only visit a small set of sites and that the adversary has samples to train on all of them [10, 16, 17, 35]. This assumption was criticized for being unrealistic [19, 29], as the world of sites that can be potentially visited is so large that not even the most powerful adversaries have the resources to collect data and train for every site. Subsequent studies have considered an open-world scenario, a more realistic setting in which the adversary can only train on a small fraction of the sites the user can visit. We use closed-world experiments for detailed comparison of different algorithms and parameter settings, and we report the results of open-world experiments for a more realistic evaluation of the attack. In the open world, we follow the terminology used in prior work: the *monitored set* includes sites that the adversary is interested in detecting, while the *unmonitored set* are all other sites.

Website vs Webpage Fingerprinting: In an abuse of language, most authors in the field use "website fingerprinting" to refer to the fingerprinting of only the home page of those websites. There is research that has attempted to fingerprint pages that are linked from the homepage [8], but virtually all studies on website fingerprinting train and test the attacks on home pages. For comparison with prior work we make same assumptions in our evaluation.

Traffic Parsing: As pointed out by Juarez et al. [19], the attacker is assumed to be able to parse all the traffic generated by a web visit and isolate it from other traffic (e.g., traffic generated by visits in other tabs, non-HTTP traffic over Tor, and so on). We note that the adversary is able to do so only if he deploys the attack from an entry node under his control. In that case, the adversary can select a domain's traffic by its Tor circuit ID. Concurrent and subsequent visits to the same domain would still go through the same circuit. If the adversary is eavesdropping the link between the client and the entry, all Tor traffic is multiplexed in the TLS connection to the entry. However, recent research has developed techniques to parse visits from multiplexed TLS traffic [40]. As with prior work, we assume that such parsing has already been done or is not needed.

3 BACKGROUND AND RELATED WORK

In this section, we categorize and summarize prior work on WF attacks and defenses and then give the necessary background on deep learning to follow the rest of the paper.

3.1 WF Attacks

Herrmann et al. were the first to evaluate WF against Tor [16]. However, they only achieved 3% accuracy in a closed world of 775 sites. The main problem with their approach was their reliance on packet length frequencies – Tor sends data in fixed-size (512-byte) packets known as cells – which renders this feature useless for classification of Tor traffic. In 2011, Panchenko et al. devised new features and improved the attack to 55% accuracy on Herrmann et al.'s dataset [28]. Since then, the success rate of WF attacks against Tor has been incrementally improved, reaching 90% accuracy by two classifiers using edit-distances [8, 39]. These attacks, however, imposed high computational costs on the adversary, which makes them impractical for real-world deployment.

Recently, a new series of WF attacks have been proposed with advanced feature sets and more sophisticated classifiers that maintain the accuracy at 90% while reducing the cost of the attack [14, 27, 38]. These attacks have become the state-of-the-art WF attacks and are used to benchmark other attacks and defenses. We have selected them in this study to compare against our deep-learning-based DF attack.

k-NN. Wang et al. [38] proposed the k-NN attack. This approach consists in applying a k-Nearest Neighbors (k-NN) classifier, including features such as packet ordering, number of incoming and outgoing cells and numbers of bursts. These features are used in combination to form a distance metric (e.g., Euclidean distance) to measure the similarity between different websites. k-NN exhibits very good performance: in a closed-world setting with 100 sites, it achieved 91% accuracy, and in an open-world setting with 5,000 sites, it achieved 86% True Positive Rate (TPR) and 0.6% False Positive Rate (FPR).

CUMUL. Panchenko et al. [27] proposed an attack based on a Support Vector Machine (SVM) classifier and devised a novel feature set based on the cumulative sum of packet lengths constructed as follows: the first coordinate in the feature vector is the length of the first packet in the traffic trace and the i-th coordinate is the sum of the value in the $(i-1)$-th coordinate plus the length of the i-th packet, where lengths for incoming packets are negative. The attack achieved 91% accuracy in a closed-world setting. In the open-world, they study two different scenarios: *multi-class*, where each monitored site is treated as a different class, and *two-class*, where the whole set of monitored pages is treated as a single class. The open world results are 96% TPR and 9.61% FPR for *multi-class* and 96% TPR and 1.9% FPR for two-class.

k-FP. Hayes and Danezis [14] proposed the k-fingerprinting attack (k-FP). k-FP uses a random forest classifier to extract fingerprints of pages: they train the random forest with traditional features, but the actual fingerprint is represented by the leafs of the trees in the random forest. The authors argue this representation is more effective for WF than the one based on the original features. To solve the open world problem, they feed these new feature vectors to a k-NN classifier. They also analyze the importance of their features and ranked them. The results show that the top 20 most important features involve counting the number of packets in a sequence, and that these leak more information about the identity of a web page than complex features such as packet ordering or packet inter-arrival time features. k-FP achieved 91% accuracy in a closed-world setting and 88% TPR and a 0.5% FPR in an open-world setting.

3.2 WF defenses

The fundamental strategy to defend against WF attacks is to add dummy packets and/or delay packets. This *cover traffic* makes WF features less distinctive, thus increasing the rate of classification errors committed by the adversary. The first defense that used this strategy against WF was BuFLO [12], proposed by Dyer et al., whose strategy was to modify the traffic to make it look constant rate and thus remove packet-specific features. However, coarse features such as total volume, size and time were hard to conceal without incurring high bandwidth overheads [12]. Tamaraw [6] and CS-BuFLO [7] tried to solve this problem by grouping sites that are similar in size and padding all the sites in a group to the greatest size in that group. Even so, these defenses still require more than 130% extra bandwidth than unprotected Tor and, on average, pages load between two to four times slower [6, 7, 12].

Recently, two lightweight countermeasures have been proposed for deployment in Tor for their low latency overhead: WTF-PAD and Walkie-Talkie.

WTF-PAD. Tor developers have expressed a preference for using *adaptive padding* as a WF defense [29, 30]. Adaptive padding [33] saves bandwidth by adding the padding only upon low usage of the channel, thus masking traffic bursts and their corresponding features. Since adaptive padding was originally designed as a defense against end-to-end timing analysis, Juarez et al. proposed

WTF-PAD, a system design for deploying adaptive padding for WF defense in Tor [20]. WTF-PAD has been shown to be effective against all state-of-the-art attacks with relatively moderate bandwidth overheads compared to the BuFLO-style defenses (e.g. 54%). Plus, since WTF-PAD does not delay packets, it does not incur any latency overhead.

Walkie-Talkie. Walkie-Talkie (W-T) has the Tor browser communicate with the web server in *half-duplex* mode, in which the client sends a request (such as for an image file) only after the server has fulfilled all previous requests. As a result, the server and the client send non-overlapping bursts in alternate directions. Moreover, the defense also adds dummy packets and delays to create *collisions*, in which two or more sites have the same features as used by the adversary's classifier. The key idea is that the traces that result from half-duplex communication can be transformed to create a collision with less padding than it would with full-duplex traces. W-T provides strong security guarantees with 31% bandwidth overhead and 34% latency overhead.

Despite the low cost of these two defenses, their evaluations have shown that each defense can significantly reduce the accuracy of the attacks to less than 30%. As of today, they are the main candidates to be implemented in Tor. In this paper, we evaluate all the attacks against them.

3.3 WF Attacks using Deep Learning

Many applications have adopted deep learning (DL) to solve complex problems such as speech recognition, visual object recognition, and object detection in images [23]. DL does not require selecting and fine-tuning features by hand. In the WF domain, there are four works that have begun to examine the use of DL.

Abe and Goto studied the application of Stacked Denoising Autoencoders (SDAE) [3] to WF attacks. They showed that SDAE is effective with 88% accuracy in the closed world and 86% TPR and 2% FPR in the open world. Although most work in deep learning recommends large data sets be used, their work was successful with only a small dataset.

Rimmer et al. proposed to apply DL for automated feature extraction in WF attacks [31]. The results show that the adversary can use DL to automate the feature engineering process to effectively create WF classifiers. Thus, it can eliminate the need for feature design and selection. In the closed-world scenario, their CNN-based attack (which we refer to as *Automated Website Fingerprinting*, or AWF) trained on 2,500 traces per site could achieve 96.3% accuracy. In their open-world evaluation, SDAE performs the best of their models with 71.3% TPR and 3.4% FPR when optimizing for low FPR. However, AWF could not outperform state-of-the-art WF attacks such as CUMUL.

Recently, Bhat et al. [5] and Oh et al. [26] have released preliminary reports on their explorations of a CNN variant and unsupervised DNNs with autoencoders, respectively. While both papers include interesting contributions, neither paper reports accuracy rates as high as those shown in our results. Additionally, neither attack was shown to be effective against WTF-PAD.

In this work we aim to bridge this gap by developing a powerful CNN-based deep learning model called deep fingerprinting (DF) that can substantially outperform all previous state-of-the art WF attacks. The DF model uses a more sophisticated variant of CNN than AWF, with more convolutional layers, better protections against overfitting, hyperparameters that vary with the depth of each layer, activation functions tailored to our input format, and a two-layer fully connected classification network. These differences in the architectural model from AWF, which are described in more detail in Section 5.3, lead to a deeper and more effective network. We show that the DF model works significantly better than AWF and all other attacks, particularly against WF defenses and in the more realistic open-world setting.

3.4 Deep Learning

In our work, we mainly focus on two deep learning techniques that previous work has shown to be promising for WF attacks.

3.4.1 Stacked Denoising Autoencoders (SDAE). Vincent et al. [37] proposed SDAEs in 2010 to improve classification performance in recognizing visual data. SDAE leverages the concept of an autoencoder (AE), a simple 3-layer neural network including input, hidden and output layers. In AE, the input data is first *encoded*, passing it through a layer of neurons to a more condensed representation (the hidden layer). The AE then performs *decoding*, in which it attempts to reconstruct the original input from the hidden layer while minimizing error. The main benefit of AE is to extract high-level features from the training data, resulting in dimensionality reduction.

A Denoising Autoencoder (DAE) uses the basic concept of AE but also adds noise to the input. The DAE tries to reconstruct the original values from the noisy inputs, which helps it to better generalize and thus handle a wider variety of inputs after training. SDAE combines ("stacks") multiple DAEs by overlapping a hidden layer as an input of the next DAE. Vincent et al. showed that SDAE achieves lower classification error rates for image classification compared to SVM, Deep Belief Networks (DBN), and Stacked Autoencoders (SAE) [37].

3.4.2 Convolutional Neural Networks (CNN). CNNs have become the gold standard in image classification after Krizhevsky et al. won the Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [22]. Schuster et al. recently proposed applying a CNN on encrypted video streams, and they show that the encrypted stream could be uniquely characterized by their burst patterns with high accuracy [32]. This suggests that CNNs could be useful for WF attacks as well. Figure 2 shows the basic architecture of a CNN [22, 24]. The architecture consists of two major components: *Feature Extraction* and *Classification.*

In *Feature Extraction*, the input is first fed into a *convolutional layer*, which comprises a set of *filters*. Each region of input is *convolved* with each filter, essentially by taking the dot product of the two vectors, to get an intermediate set of values. These values are input to an *activation function* – this is similar to neurons being activated based on whether or not the filtered input has certain features. Having more filters means being able to extract more features from the input. The output of the activation function is then fed into a *pooling* layer. The pooling layer progressively reduces the spatial size of the representation from the feature map to reduce the number of parameters and amount of computation. The most

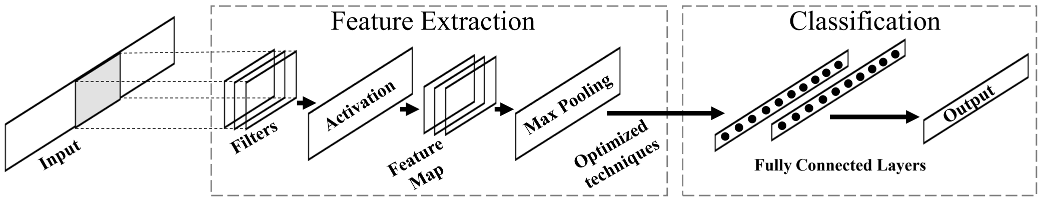


Figure 2: A basic architecture of convolutional neural networks (CNN)

common approach used in pooling is *Max Pooling*, which simply selects the maximum value in a spatial neighborhood within a particular region of the feature map to be a representation of the data. This has the advantage of being invariant to small transformations, distortions and translations in the input, since the largest signals in each neighborhood are retained. The final part of the feature extraction component (*Optimized techniques* in Figure 2) mainly consists of a stochastic *dropout* function and *Batch Normalization* that help improve classifier performance and prevent overfitting.

The CNN then passes the output from the convolutional and pooling layers, which represents high-level features of the input, into the *Classification* component. In this component, a set of fully-connected layers uses the features to classify the input. During training, the loss value of classification is used to not only update weights in the classification component but also the filters used in feature extraction. To estimate the loss value, we use *categorical cross-entropy*, which is suitable for multi-class classification problems such as WF.

4 DATA COLLECTION

For the closed-world dataset, we visited the homepage of each of the top Alexa 100 sites 1,250 times and dumped the traffic generated by each visit separately using `tcpdump`. We used ten low-end machines in our university's campus to collect the data. We have followed prior work's methodology for data collection [19, 39]; on each machine, the visits were sequential and were ordered according to Wang and Goldberg's batched methodology to control for long- and short-term time variance [39]. More specifically, we split the visits to each site in five chunks, so that the websites are accessed in a round-robin fashion: in each batch we access *each* site 25 times. As a result of batching, the visits to a site are spread over time. The rationale for this is twofold: i) to avoid having our IP addresses banned by the web servers; and, ii) to capture variants of the sites over time for more robust training and testing.

We used `tor-browser-crawler` [19] to drive the Tor Browser to visit websites. This allows for more realistic crawls than using tools like `wget` or `curl` because the setting resembles a real user browsing the Web with Tor. We acknowledge that to be more realistic, our crawler should model user browsing behavior when crawling sites. However, modeling user behavior in Tor is challenging, as user statistics are not collected for privacy reasons. Virtually all existing datasets collected for WF follow the same simplistic user model we use in this study.

After the crawls were finished, we discarded corrupted traffic traces. For instance, we removed traces that did not have any incoming or outgoing packets or were too short – less than 50 packets. After removing corrupted traces, we only kept the sites, or classes, that had at least 1,000 visits. We ended having 95 sites with 1,000 visits for our closed-world evaluations. We refer to the set of the data used for closed-world evaluations as the *closed-world* dataset.

Open-world dataset. For the open-world dataset, we visited the sites from Alexa's top 50,000, excluding the first 100 sites used to build the closed-world dataset. We used the same ten machines to collect the data, where each machine collected the data for 5,000 different sites sequentially. We visited each open-world site only once and took a screenshot of their homepages. After collecting the data, we discarded corrupted visits the same way we did for the closed-world dataset. During the crawling of the open-world, we found sites returning an access denied error message, a timeout error, or a blank page. Moreover, many of the sites were behind Cloudflare's CDN, which presents a CAPTCHA to connections coming from Tor exit relays. We removed those sites from our dataset by comparing their homepage's screenshot with each of: a blank page, an access denied page, a CAPTCHA page, and a timeout error page. The final dataset has a total of 40,716 traffic traces.

Defended dataset. To evaluate the defenses, we produced datasets with traces protected by each defense: for BuFLO, Tamaraw and WTF-PAD, we protect traces by padding them according to the defense protocols, using the scripts and simulators provided by the authors [6, 12, 20]. Walkie-Talkie, however, cannot be completely simulated, as half-duplex communication is hard to model. We thus performed a new crawl with a Tor Browser in half-duplex mode. Since the implementation of half-duplex for the original implementation of Walkie-Talkie was done in an outdated version of the Tor Browser, we had to implement half-duplex in the latest version of Tor Browser at the time of our crawls (Tor Browser Bundle version 7.0.6). With this modified Tor Browser, we collected closed- and open-world datasets of size similar to the undefended ones. Walkie-Talkie also requires padding the bursts in the half-duplex traces. To that end, we followed the *mold* padding strategy as described in the Walkie-Talkie paper [41].

5 EXPERIMENTAL EVALUATION

In this section, we evaluate WF attacks based on SDAE, DF and AWF. We compare them with the state-of-the-art WF attacks. We used our datasets for these evaluations.

Table 1: Hyperparameters selection for DF model from Extensive Candidates Search method

Hyperparameters	Search Range	Final
Input Dimension	[500 ... 7000]	5000
Optimizer	[Adam, Adamax, RMSProp, SGD]	Adamax
Learning Rate	[0.001 ... 0.01]	0.002
Training Epochs	[10 ... 50]	30
Mini-batch Size	[16 ... 256]	128
[Filter, Pool, Stride] Sizes	[2 ... 16]	[8, 8, 4]
Activation Functions	[Tanh, ReLU, ELU]	ELU, ReLU
Number of Filters		
Block 1 [Conv1, Conv2]	[8 ... 64]	[32, 32]
Block 2 [Conv3, Conv4]	[32 ... 128]	[64, 64]
Block 3 [Conv5, Conv6]	[64 ... 256]	[128, 128]
Block 4 [Conv7, Conv8]	[128 ... 512]	[256, 256]
Pooling Layers	[Average, Max]	Max
Number of FC Layers	[1 ... 4]	2
Hidden units (each FCs)	[256 ... 2048]	[512, 512]
Dropout [Pooling, FC1, FC2]	[0.1 .. 0.8]	[0.1, 0.7, 0.5]

5.1 Implementation

Our implementation of the DF model uses the Python deep learning libraries *Keras* as the front-end and *Tensorflow* as the back-end [1]. The source code of the implementation and a dataset to reproduce our results is publicly available at https://github.com/deep-fingerprinting/df.

5.1.1 Data Representation. In WF, a website trace is represented as a sequence of tuples <$timestamp$, ±$packet\_size$>, where the sign of $packet\_size$ indicates the direction of the packet: positive means outgoing and, negative, incoming.

Prior work in WF has shown that the most important features are derived from the lengths of traces in each direction [14, 38]. Wang et al. [38] simplified the raw traffic traces into a sequence of values from $[-1, +1]$, where they ignored packet size and timestamps and only take the traffic direction of each packet. However, we performed preliminary evaluations to compare the WF attack performance between using packet lengths and without packet lengths, i.e., only packet direction, as feature representations. Our result showed that using packet lengths does not provide a noticeable improvement in the accuracy of the attack. Therefore, we follow Wang et al.'s methodology and consider only the direction of the packets.

SDAE, DF and AWF require the input to have a fixed length. In order to find the input length that performs best, we parameterized it and explored the range $[500,\ 7,000]$, which contains most of the length distribution in our data. Our results show that 5,000 cells provide the best results in terms of classification accuracy. In practice, most of the traces are either longer or shorter than that. We padded shorter traces by appending zeros to them and truncated longer traces after 5,000 cells. Out of 95,000 traces in the closed-world dataset, only 8,121 were longer than 5,000 cells and had to be truncated, while the rest were padded.

5.1.2 SDAE. We reproduced Abe and Goto's results [3], as described in Section 3. Following guidance from the authors, we successfully re-implemented their SDAE neural network on the same architecture and dataset they used. We achieved 89% accuracy, a slightly higher accuracy than the one Abe and Goto reported in their paper. We believe that using different Python DL modules (*Kera* and *Tensorflow*) and randomly initializing the weights accounts for this difference. Furthermore, we slightly changed their SDAE model's hyperparameters to improve its performance in our experiments.

5.1.3 AWF. Rimmer et al. provided us with their source code to re-produce their results. We strictly followed their proposed hyperparameters and evaluate the model in our dataset to make a fair comparison for our model and the previous state-of-the-art WF attacks.

5.1.4 DF. To develop our DF model to effectively perform WF attacks on both non-defended and defended dataset, we have followed techniques in the deep learning literature [21, 22, 36] to improve the performance of the model, such as using the appropriate number of convolutional layers, mitigation and prevention of overfitting [34] and a suitable activation function for our WF input data [25]. These studies helped us design the sophisticate architecture and tune the model's hyperparameters that best fit for WF.

We adapted the base CNN model of DF to our needs, as there are important differences between traffic analysis and traditional applications of CNN-based models such as image recognition. For example, standard activation functions such as *sigmoid* and *rectified linear unit (ReLU)* do not activate on negative values and thus will not use the information conveyed by the sign of the input (i.e., cell direction). Activation functions that can handle negative inputs include tanh, leaky ReLU (LReLU), parametrized ReLU (PReLU) and Exponential Linear Unit (ELU). Prior work has shown that ELU provides fast and accurate classification [11, 25]. We compared ELU with other activation functions during hyperparameter tuning and it performed the best among all the functions we tested (see Section 5.2). Although the traditional tanh function can also handle negative inputs, it is vulnerable to the *vanishing gradient* issue [4], which slows down optimization.

Another difference between traffic and image data is that images are two-dimensional, whereas our data is a vector. This means that filter and pool sizes cannot be two-dimensional (e.g., 2x4) but have to be cast to one dimension (e.g., 1x4).

5.2 DF's Hyperparameter Tuning

A fundamental process in supervised classification is to tune the *hyperparameters* of the classification model, such as the kernel used in an SVM or the number of hidden layers in a CNN. This process involves adjusting the trade-off between variance, bias and classification performance, so that the model fits the training data while still generalizing to samples that it has not been trained on. For DF, however, the large amount of training data and the large number of hyperparameters the model has render an exhaustive search prohibitive in terms of computational resources. To demonstrate our attacks, we thus only aim at a good-enough classifier and we acknowledge that someone with more resources might be able to optimize our model further.

To select the hyperparameters for our models, we perform an extensive search through the hyperparameter space, in which we build each layer of the deep learning model block by block. In each building block, we vary the hyperparameters to estimate the gradient of the parameter and determine whether we must increase or decrease its value. Once this process is done, we select the best top-n parameters and use them as the initial parameters for the optimization in the next block. When all layers are set, we select the best combination of hyperparameters.

By the transferability property of neural networks [42], WF attacks based on other models can use the values we found for the DF hyperparamters to bootstrap the hyperparameters of their model. We thus used the transferability property to find the parameters for the defended datasets from the hyperparameters found using the undefended dataset. We only needed to slightly adjust some hyperparameters to optimize our model, significantly reducing the time spent in hyperparameter tuning. We thoroughly illustrate and explain our design of DF model in Appendix A. The search space as well as the final selected values are shown in Table 1.

Evaluating overfitting Even though deep neural networks (DNN) are a powerful supervised classification model, they are, as with as most machine learning models, vulnerable to overfitting. Overfitting occurs when the model errs on samples that it has not been trained on, meaning that the model cannot generalize. For small datasets, overfitting can be measured with cross-validation techniques. For large datasets like ours, however, we can just split the data into three mutually exclusive sets: training, validation and testing, with a ratio of 8:1:1. We then measure the difference in error rate between different sets. In case of overfitting, the model would achieve a substantially higher accuracy on the training set than on the testing set.

During the training of our DF model, we applied *Dropout* [34] and *Batch Normalization (BN)* [18] to prevent overfitting. These are regularization techniques that loosen the model and allow for greater generalization. In dropout, the model randomly selects hidden units, including their incoming and outgoing connections, and temporarily removed them from the network while training. BN normalizes the fully-connected and convolutional layers' outputs and helps accelerate learning while also reducing overfitting. Moreover, we analyze the error rates between training and testing datasets during hyperparameter tuning to ensure that our model is not overfitting.

Figure 4 depicts the training and testing error rates. The difference between training and testing error of the DF model is less than 2%, suggesting that overfitting is unlikely.

5.3 Differences Between DF and AWF

We now explain the significant differences of the DF model compared to the AWF model proposed by Rimmer et al. [31] that help explain the superior performance of DF.

Basic Block Design. The basic block is the group of convolutional layer(s), max pooling layer, filters and activation layer(s) that perform feature extraction in a CNN. Generally, the basic block is repeatedly appended to create deeper networks.

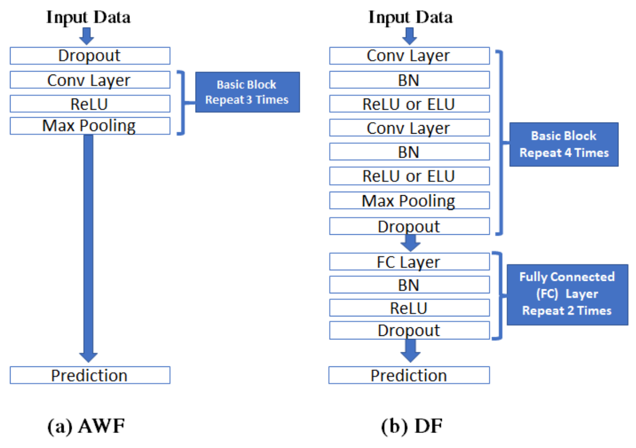


Figure 3: Comparison between AWF and DF models

We observe that the AWF model is similar to Imagenet [22], one of the earliest CNN models proposed in 2012. The basic block of this model only contains one convolutional layer followed by one max pooling layer as shown in Figure 3(a). In contrast, our DF model is inspired by modern large image classification networks such as VGG [21], GoogleNet [36] and ResNet [15] that apply at least two consecutive convolutional layers before a max pooling layer as shown in Figure 3(b). Max pooling typically reduces the data to a smaller size, so it is not possible to have deeper networks when pooling after every convolutional layer. Adding more convolutional layers in each basic block thus enables more convolutional layers in total and a deeper network with more effective feature extraction.

Overfitting Concerns. Rimmer et al. criticized the CNN model for having a higher risk of overfitting which was shown in their experimental results. We argue that a more carefully crafted model can mitigate overfitting. The AWF model includes a dropout layer before the first basic block as shown in Figure 3(a). While dropout is a common technique to help prevent overfitting, this placement is atypical in CNN designs, as it may result in the loss of meaningful features extracted from the input and may not be sufficient to prevent overfitting. In DF, we used overfitting-contention mechanisms that are applied in the state-of-the-art CNN networks for computer vision, including a batch normalization (BN) layer that is added right after each convolutional layer and a dropout layer after the activation function, as explained in Section 5.2. With these mechanisms, the DF model shows no evidence of overfitting in our experiments.

Varying Hyperparameters In the AWF model, the value of some hyperparameters are fixed such as using 32 filters in every convolutional layer. Using a fixed number of filters over all the layers reduces the capability of the model to learn. In contrast, the DF model follows the state-of-the-art in computer vision by varying hyperparameter values for different layers [21]. For example, we increase the number of filters as we get deeper in the network. The intuition behind varying the values is that the CNN uses hierarchical features in its processing pipeline. The features in lower layers (close to the input) are primitive, like edge detection, while features in upper layers are high-level abstract features, like object detection, made from combinations of lower-level features. We increase the number of filters at higher layers to improve the ability to encode richer representations.

Activation Function. The AWF model only uses the ReLU activation function. ReLU is popular in CNNs, but it maps all negative values to zero. Our input formats include negative values that represent incoming packets, so using ReLU in convolutional layers close the input can substantially reduce the information available to deeper layers in the model. In the DF model, the activation function in the first basic block is ELU, which can learn a data representation containing negative values to ensure that the model can learn all meaningful representations from the input.

Fully-connected Layers. The AWF model directly connects the last max pooling layer to the prediction layer, a densely connected layer with an output size equal to number of classes. In more recent CNNs, there are a set of fully connected (FC) layers that follow the convolutional layers and precede the prediction layer. The FC layers play an important role in the learning and classification processes of the model. Essentially, the convolutional layers perform feature extraction, but that means that it is important to carefully design the classifier that uses the extracted features, which is the role of the FC layer. In the DF model, we add two FC layers with the combination of BN and dropout to prevent the overfitting that normally occurs in FC layers.

Overall, our DF model was specifically designed to effectively perform WF attacks by leveraging the state-of-the-art techniques from computer vision research. We provide a thorough explanation on how the DF model was developed and a visualization of the DF model in Appendix A to allow other researchers to gain better understanding and reproduce our work. Our experimental results confirm that DF performs better than AWF model in defended and non-defended and on both closed-world and more realistic open-world scenarios. These results help to illustrate the impact of the DL architecture design on the performance of the attacks.

5.4 Closed-world Evaluation on Non-defended Dataset

We evaluate the performance of the DF attack in the closed-world scenario on the *non-defended* dataset, which comprises website traces from the *closed-world* dataset with no WF defenses in place. Moreover, we compare DF model with the state-of-the-art WF attacks: *k*-NN, *CUMUL*, *k*-FP, AWF, and SDAE. We re-evaluate these attacks on our *non-defended* dataset and apply *k-fold cross-validation* for training their classifiers and testing their performance, as was done in the papers presenting these attacks [14, 27, 38].

Table 2: Closed World: Accuracy on the non-defended dataset for state-of-the-art attacks.

Classifier	SDAE	DF	AWF	*k-NN*	CUMUL	*k-FP*
Accuracy	92.3%	**98.3%**	94.9%	95.0%	97.3%	95.5%

Table 2 shows the accuracy results. Our DF model attains 98.3% accuracy, which is better than the other attacks and higher than any previously reported result for a WF attack in Tor. Our DF model

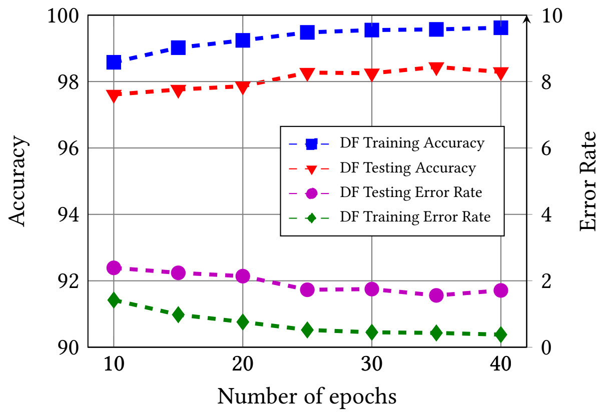


Figure 4: Closed World: Impact of the number of training epochs on DF accuracy and error rate

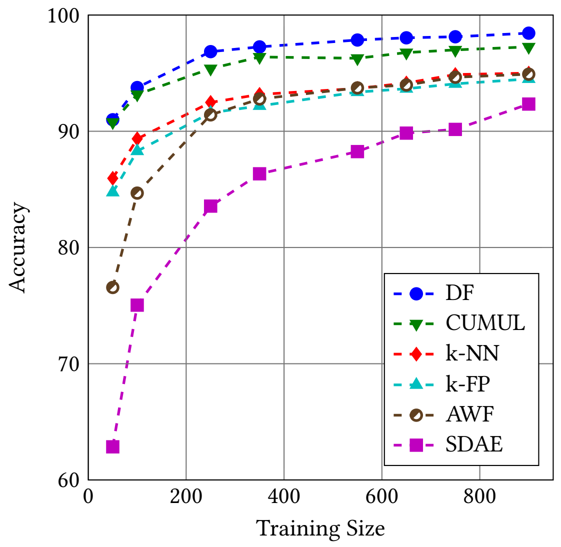


Figure 5: Closed World: Impact of numbers of training traces on classification accuracy

performs better than AWF. Our results for AWF (94.9%) are a bit lower than the reported results by Rimmer et al. (96.5%), we believe this is due to the larger dataset used by them. We observe that CUMUL, *k*-FP, *k*-NN and SDAE benefit from the larger training data set with 2-4% higher accuracies than previously reported results that used smaller datasets (usually 90 training instances). SDAE was not as accurate as the other attacks.

Additionally, we investigate how fast the model can learn to distinguish patterns from the input data, also known as *convergence* of the model. This depends on many factors such as the method used for data pre-processing, the DL architecture and the hyperparameters used to create the model. This evaluation helps to validate the quality of our hyperparameter tuning method. Moreover, the attacker can use this to estimate the number of *training epochs*, rounds of feeding training data into the classifier, required for the classifier to reach the expected level of accuracy. Normally, the classifier gradually learns better with more training epochs.

Figure 4 shows that with only 10 training epochs, DF can reach testing accuracy of about 97%, DF consistently improves with more training epochs, until accuracy levels off after 30 epochs.

Finally, we investigate the impact of dataset size on classifier accuracy. The results shown in Figure 5 indicate that DF and CUMUL consistently outperform the other attacks for all training sizes. With just 50 traces per site, both DF and CUMUL achieve 90% accuracy. k-NN, k-FP and AWF require 250 traces to reach this accuracy, and SDAE requires 750 traces. The observed accuracies mostly saturate after 550 traces, except for SDAE. The results show that the various techniques used in the DF model lead to significantly better performance compared to the simpler AWF model.

5.5 Training Cost

We now examine the training time for WF attacks using DL in comparison to state-of-the-art WF attacks. We found that with GPU acceleration by using NVIDIA GTX 1070 with 8 GB of GPU Memory, SDAE required 16 minutes for training (13 minutes for pre-training and 3 minutes for fine-tuning processes), DF required 64 minutes for 30-epoch training. The relatively simpler AWF model requires 4 minutes for 30-epoch training. Without a GPU, SDAE required 96 minutes, DF required approximately 10 hours, and AWF required 1 hour. For training the other attacks, we found it required 12.5 hours for k-NN, 57 hours for CUMUL (parallelized with 4 processes), and 1 hour for k-FP. Overall, SDAE, DF and AWF have reasonable training times, particularly when using a GPU.

5.6 Closed-world Evaluation on the Defended Dataset

We next examine the performance of WF attacks against Tor traffic with defenses in the closed-world scenario. It is important to note that the attacker needs to train the classifiers with defended datasets to perform this attack. As mentioned in Section 3, several WF defenses have been proposed that they can reduce the accuracy of state-of-the-art WF attacks to less than 50%. Notably, WTF-PAD and Walkie-Talkie offer both effective protection and reasonable overheads, such that they are realistic for adoption in Tor. With our larger dataset, we conduct an evaluation on SDAE, DF, AWF and prior attacks against these defenses, as well as BuFLO and Tamaraw.

Table 3 shows the overheads of each defense and the accuracy of the attacks against defended datasets. BuFLO and Tamaraw, the two high-overhead defenses, hold up well with less than 17% accuracy. The attacks also manage at most 49.70% accuracy against Walkie-Talkie due to symmetric collisions. A surprising result is that DF achieves over 90% accuracy against WTF-PAD. Our tests of WTF-PAD showed 64% overhead, which means that there was more padding on average than in the Juarez et al.'s study [20], and yet the attack was successful. More generally, it seems that the larger amount of traces per site compared to the original WTF-PAD evaluation has played a role in the higher accuracy attained by the attack. For example, k-FP achieved nearly 69% accuracy in our experiment, whereas Hayes and Danezis tested k-FP against their own implementation of adaptive padding and obtained 30% accuracy [14].

DF significantly outperforms AWF on the dataset protected by WTF-PAD, with a much larger gap in performance than observed on the undefended dataset. We believe that the deeper network is able to better extract useful features in the WTF-PAD data that the AWF model is unable to find, leading to this result. The model architecture in DF plays a key role in its flexibility to generalize to defended traffic.

We note that the overheads for BuFLO and Tamaraw are higher than reported in prior work at 246% and 328% bandwidth overheads, respectively. Furthermore, we found that the larger the dataset, the greater the packet timing variance is, which is fundamental to determine the padding rate. Also, Tamaraw has higher overheads than BuFLO, which contradicts the purposed intended with its design and the overheads reported in previous evaluations. The cause of this is a greater amount of padding after the transmission has finished in Tamaraw compared to BuFLO. BuFLO stops padding immediately after the transmission has finished, as long as the transmission has lasted for longer than ten seconds, which is the case for most of the traces in our dataset.

With such heavy overheads, BuFLO and Tamaraw are not practical to deploy as a WF defense in Tor. WTF-PAD and Walkie-Talkie have lower overheads, and Tor Project developers have already shown an interest in deploying *adaptive padding* as a possible defense [29, 30]. We thus select WTF-PAD and Walkie-Talkie for our open-world evaluation.

5.7 Open-world Evaluation

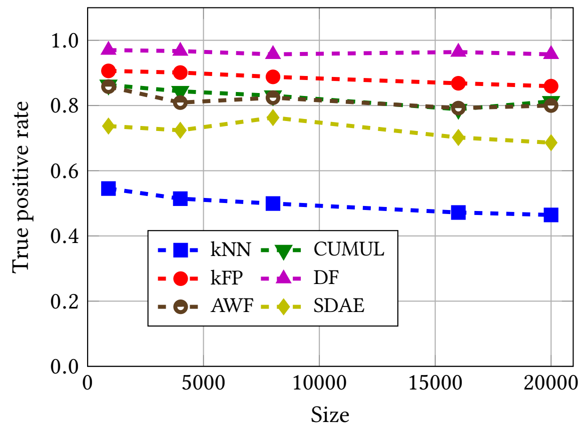
We now evaluate the performance of the attack in the more realistic open-world setting. As mentioned in Section 2, in the open-world scenario, the adversary not only classifies traffic traces based on a limited set of monitored sites, but he must also distinguish whether the trace comes from a monitored site or an unmonitored one.

In our evaluation, we assess the performance of classifiers in the open-world scenario on each model by showing true positive rate (TPR) and false positive rate (FPR), but also with precision and recall curves, recommended in the WF literature [20, 27] as more appropriate metrics for the open-world evaluation than TPR and FPR. The size of the monitored and unmonitored sets are heavily unbalanced, so using only TPR and FPR can lead to incorrect interpretation due to the base-rate fallacy. We also provide ROC curves in Appendix C.

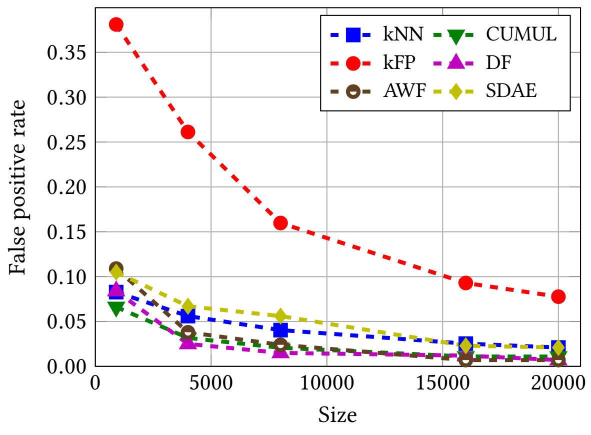
Standard Model. In previous studies on WF in the open-world setting [14, 27, 38], it has been assumed that if the attacker included unmonitored traces when training the classifier, it could help the classifier better distinguish between monitored and unmonitored traces. This assumption is also common in machine learning, and we thus call it the Standard model. Fundamentally, the process of training and creating datasets used during open-world evaluation is the same as in the closed-world scenario, except that we additionally train on unmonitored websites traces as another class. To investigate the impact of more training data on the performance of the classifiers in the open-world scenario, we train the classifier with different portions of the unmonitored dataset.

Table 3: Accuracy in a closed-world scenario on defended datasets, SDAE, DF, and AWF vs. the state-of-art WF attacks

Defenses	Overhead		Accuracy of WF attacks on defended datasets					
	Bandwidth	Latency	SDAE	DF	AWF	*k-NN*	CUMUL	*k-FP*
BuFLO	246%	137%	9.2%	12.6%	11.7%	10.4%	13.5%	13.1%
Tamaraw	328%	242%	11.8%	11.8%	12.9%	9.7%	16.8%	11.0%
WTF-PAD	64%	0%	36.9%	**90.7%**	60.8%	16.0%	60.3%	69.0%
Walkie-Talkie	31%	34%	23.1%	49.7%	45.8%	20.2%	38.4%	7.0%



(a) TPR



(b) FPR

Figure 6: Open World: The impact of the amount of unmonitored training data on TPR and FPR (Non-defended dataset).

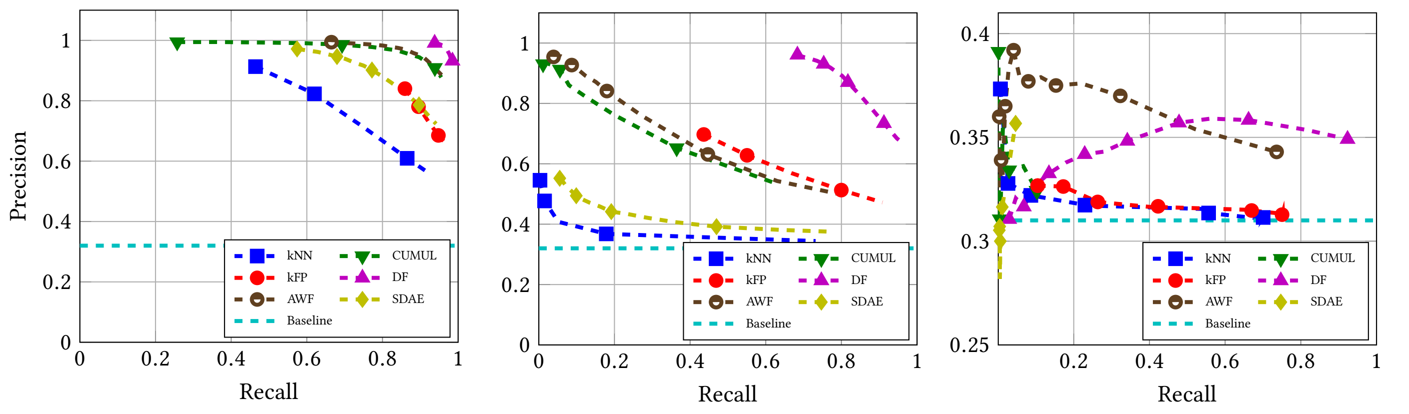
In our open-world evaluation, we use the prediction probability to classify the input traces. In particular, if the input trace is a monitored website trace and the maximum output probability belongs to any monitored site and is greater than a threshold, we consider this as a *true positive*. We used different thresholds for different WF attacks. We selected the thresholds for each WF attack such that they have high TPR and low FPR. Figure 9 in Appendix C shows examples of ROC curves for WF attacks against Non-defended, WTF-PAD, and W-T datasets. Following the experimental procedures of Rimmer et al. [31] and Panchenko et al. [27], we focus on the binary results of whether the input trace is classified as monitored (predicted to be in any of the monitored classes) or unmonitored. Note that if the trace is determined to be monitored, the attacker can then use the multi-class classification to predict which website the user has actually visited.

k-NN and *k*-FP attacks use the *k-nearest neighbors* algorithm in their predictions and do not output the probability of predictions. For these attacks, we consider the prediction probability of a site as the fraction of the nearest neighbors belonging to that site among the k nearest neighbors. We explored the performance of these attacks as the value of k varies from 2 to 10. We found that above $k = 5$, the TPR and FPR do not change significantly. For our open-world evaluation, we used $k = 6$ in both k-NN and k-FP.

5.7.1 Results. We first evaluate efficacy of our WF attack in the Standard model as amounts of unmonitored training data varies and compare it with other state-of-the-art WF attacks on the non-defended traces. Our training set in this experiment contains 85,500 monitored traces (900 instances for each of 95 monitored sites) and we varied the number of unmonitored sites from 900 to 20,000 sites (one instance for each). Our testing set includes 9500 monitored traces (100 instances for 95 monitored sites) and 20,000 unmonitored traces (one instance for 20,000 unmonitored sites). Note that the 20,000 unmonitored sites in the testing are different from those in the training.

As shown in Figures 6a and 6b, the TPR tends to slightly decrease with the reduction of FPR as the size of unmonitored training data increase for all the WF attacks. The results show that the DF model consistently performs best on both TPR and FPR, with 0.957 TPR and 0.007 FPR for 20,000 unmonitored training sites. k-NN has the lowest TPR and k-FP has the highest FPR. The DF, CUMUL and AWF have the same FPR trend as the training size increases, but DF has higher TPR than CUMUL and AWF over all the training sizes.

Our results show that as we increase the size of the unmonitored class in the training, the FPR drops and it reaches its lowest amount at size 20,000. In the next experiment, we fix the number of training samples for the unmonitored class to 20,000 and we evaluate the diagnostic ability of WF attacks as the discrimination threshold is varied. We next perform the experiment on our non-defended, WTF-PAD and Walkie-Talkie (W-T) datasets. As mentioned in Section 4, for W-T, we cannot use the same dataset to W-T traces as it required to be directly captured from half-duplex connections from Tor browser. Our training set for the W-T evaluation contains 91,000 monitored traces (910 instances for each of 100 monitored sites) and we varied the number of unmonitored sites from 900 to 20,000 sites (one instance for each). Our testing set includes 9,000 traces (90 instances for each of 100 monitored sites) and 20,000 unmonitored



(a) Non-defended dataset **(b) WTF-PAD dataset** **(c) W-T dataset**

Figure 7: Open World: Precision-Recall curves.

traces (one instance for 20,000 unmonitored sites). In the following open-world experiments, we mainly focus on the precision and recall to avoid the base-rate fallacy as mentioned above.

Figure 7 shows the precision-recall curves for WF attacks in our non-defended, WTF-PAD and W-T datasets. Precision-recall curves are used to represent the performance of the classifier as an alternative to ROC curves in imbalanced datasets. Imbalanced datasets have an impact on precision, an important metric to measure performance, however, ROC curves do not take precision into account. This choice is specially relevant in the open-world evaluation, as the size of the monitored set is typically orders of magnitude smaller than the unmonitored set and as such it should be represented in our testing set, thus leading to an imbalance [20].

As we see in the figure, the DF attack outperforms the other state-of-the-art WF attacks in all three cases. In the non-defended dataset, it is highly effective for any threshold. The CUMUL and AWF attacks in Figure 7a have high precision but a very wide range of recall, which means the attacks miss many monitored visits. For traffic defended by WTF-PAD, Figure 7b shows a reduction of both precision and recall for all WF attacks. The DF attacker does the best. Tuned for high precision, it achieves precision of 0.96 and recall of 0.68. Tuned for high recall, it reaches 0.67 precision and 0.96 recall. All the other WF attacks get close to the baseline (random guessing) as the threshold decreases. The result shows that the otherwise robust WTF-PAD is significantly undermined by the DF attack.

Figure 7c shows the precision-recall curves for the W-T dataset. The attacks all perform quite poorly, with all except the DF attack close to the baseline. The DF attack does moderately better but still has a precision of less than 0.36 in all cases.

5.8 A Deeper Look at W-T

Top-N prediction for closed-world W-T Wang and Goldberg explain that any attack using the main features of the state-of-the-art attacks can get at most 50% accuracy against W-T [41]. In our closed-world results, the DF attack nearly reached this theoretical maximum. We now examine prediction probability for DF against W-T. We consider top-N prediction, in which we look at not only the highest probability (Top-1 prediction), but also the top N probability values. Surprisingly, we only need to look at the case of $N = 2$. Top-2 prediction accuracy reaches 98.44% accuracy. This likely means that DF is correctly selecting the real site and the decoy site and, as expected, having to guess randomly between them. We discuss the importance of this result in Section 6.

Asymmetric Collision (Closed-World) W-T requires that the client create symmetric collisions between pairs of websites (site A is molded with site B and vice versa). Since this requires storing all the pairs, a simpler implementation would ignore this requirement and have the client random select the decoy site for each access, resulting in asymmetric collisions. In this setting, the DF attack is much more accurate at 87.2%, compared to 49.7% with symmetric collisions. This shows the importance of creating symmetric collisions in W-T.

Asymmetric Collision (Open-World) We next evaluate the scenario that 10% of the users do not follow the W-T guidelines, in that they visit a non-sensitive site and choose a non-sensitive site as a decoy instead of a sensitive one, and when they visit a sensitive site they choose a sensitive site as a decoy instead of a non-sensitive one. In this scenario, TPR is increased to 0.85 TPR, and FPR is significantly reduced to 0.23 TPR, compared to the case that all users strictly follow the procedure to create symmetric collisions which has 0.80 TPR and 0.76 FPR. Thus, the major goal of W-T to create the confusion between sensitive websites and non-sensitive websites could be undermined in some scenarios.

6 DISCUSSION

The results of our study show that deep learning, and the DF approach in particular, is a powerful tool for WF attacks against Tor. Further, our results against defended traffic show that WTF-PAD is potentially vulnerable against deep-learning-based WF, with high precision even in the open-world setting. Based on what we have observed during experimental evaluations, we now discuss several new directions in both improving the attacks and exploring designs for more effective defenses.

Improving Open-World Classification. In our study, we observed that designing the CNN architecture and tuning hyperarameters are specific to both the environment and input data. For example, the gap in performance between DF and AWF was much larger for the open-world setting than the closed world. Additional exploration of models in the open-world scenario, such as the depth and number of convolutional layers, different filter sizes, or different dropout parameters, may yield improved results beyond what we have found so far. More training data may also help the classifier better distinguish between monitored and unmonitored pages. Our simple data format might be extended to include, for example, statistical timing information that is currently excluded.

Finally, we note that the attacker can perform a targeted attack on users in a *semi-open-world* setting, in which the targeted users can be profiled as likely going to a subset of sites. For example, if the user is known to only read one or two languages, then many sites in other languages can be eliminated from the set. Alternatively, a user's profile can help the attacker identify some likely sites for her interests, such that the classification of statistically similar monitored sites may be dismissed as likely false positives.

Attack Costs. The time and effort needed to collect and train on large data sets can have practical implications for weaker attackers. Collecting large data sets as used in our experiments requires multiple PCs running for several days. Both Juarez et al. [19] and Wang and Goldberg [40] show that after 10-14 days, the accuracy of WF attacks goes down significantly. A weak attacker might need to choose between limiting the scope of monitored sites, living with the results of using stale and inaccurate data, or using fewer training samples per site. We note that, even though deep learning works best with more data, DF performs well in the closed-world setting even with smaller datasets. Additionally, we found that while k-FP, AWF, and DF can be trained quickly on large datasets, k-NN and CUMUL do not scale well to larger data. In particular, due to hyperparameters grid search, CUMUL took us days to train. Further exploring the trade-offs between scalability and accuracy remain important areas for future research.

WTF-PAD. As DF can break WTF-PAD with over 90% accuracy in the closed-world setting, we now consider why the defense failed by examining the adaptive padding algorithm at the heart of WTF-PAD. Adaptive padding aims to detect large timing gaps between bursts and use padding to make these gaps less distinctive. While Juarez et al. showed that this is effective against prior WF attacks [20], DF can still detect patterns that remain after WTF-PAD is applied. When used in analyzing images, CNN can detect an object (e.g. a dog) anywhere in an image due to its use of convolutional layers with multiple filters. Similarly, DF can detect any small region or portion of distinguishing patterns, no matter where those patterns are located in the trace. Adaptive padding only randomly disrupts some patterns in the trace, leaving other patterns relatively unperturbed.

Walkie-Talkie. Walkie-Talkie (W-T) has an advantage over WTF-PAD, as it focuses directly on features used in WF attacks, and it seeks explicitly to create collisions. Indeed, W-T performed much better than WTF-PAD against DF, which would seem to make it a strong candidate for Tor. We note, however, that there are several downsides to deploying W-T that require further investigation to overcome:

- It requires the directory server to collect and distribute to all clients a database of website patterns that can be used to set the padding patterns. The patterns need to be kept up to date to provide effective plausible deniability.
- Half-duplex communication adds to the latency of fetching a site, 31% according to Wang and Goldberg [41], which is a direct cost to end-user performance in a system that is already slower than regular browsing.
- According to Wang and Goldberg, the browser is expected to pair sensitive and non-sensitive pages and, ideally, pay attention to issues such as language to select realistic cover pages. To be most effective, then, the browser has to have a lot of context about the user and the nature of her activity, which is hard to build into the system.
- Given that DF achieves very high Top-2 accuracy, the attacker can use auxiliary information such as language to guess the real site. Further, if the system does not assign a decoy site to a particular sensitive site or page (e.g. beyond the homepage of the site), then that site is effectively uncovered, because it will not be used as a decoy for any non-sensitive sites.

Alternative Defenses. To improve effectiveness against DF without requiring extensive interaction with the browser, defenses could apply *adversarial machine learning* [9, 13] to generate the adversarial website traces to confuse the classifier. This is challenging to do compared to adversarial machine learning in image processing tasks, since the web trace is happening live, where the Tor client does not know the full picture of the trace in advance. Further, Tor is limited in how it can manipulate the trace—it can add dummy packets and delay packets but not delete packets or speed them up. Addressing these challenges would be interesting for future work.

7 CONCLUSION

In this study, we investigated the performance of WF using deep learning techniques in both the closed-world scenario and the more realistic open-world scenario. We proposed a WF attack called Deep Fingerprinting (DF) using a sophisticate design based on a CNN for extracting features and classification. Our closed-world results show that the DF attack outperforms other state-of-the-art WF attacks, including better than 90% accuracy on traffic defended by WTF-PAD. We also performed open-world experiments, including the first open-world evaluation of WF attacks using deep learning against defended traffic. On undefended traffic, the DF attack attains a 0.99 precision and a 0.94 recall, while against WTF-PAD, it reaches a 0.96 precision and a 0.68 recall. Finally, we provided a discussion on our results along with suggestions for further investigation.

Overall, our study reveals the need to improve WF defenses to be more robust against attacks using deep learning, as attacks only get better, and we have already identified several directions to improve the DF attack further.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful feedback. A special acknowledgement to Vera Rimmer for providing feedback that helped improve the paper. We appreciate the interesting discussions with Vera Rimmer, Dr. Leon Reznik and Igor Khokhlov that helped developing this paper.

This material is based upon work supported by the National Science Foundation under Grants No. CNS-1423163 and CNS-1722743. In addition, this work has been supported by the European Commission through KU Leuven BOF OT/13/070 and H2020-DS-2014-653497 PANORAMIX. Juarez is supported by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO).

REFERENCES

[1] 2017. Keras. https://keras.io/. (2017).

[2] 2017. Users - Tor metrics. https://metrics.torproject.org/userstats-relay-country.html. (2017).

[3] K. Abe and S. Goto. 2016. Fingerprinting attack on Tor anonymity using deep learning. In *in the Asia Pacific Advanced Network (APAN)*.

[4] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (Mar 1994), 157–166. https://doi.org/10.1109/72.279181

[5] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. 2018. Var-CNN and DynaFlow: Improved Attacks and Defenses for Website Fingerprinting. "https://arxiv.org/pdf/1802.10215.pdf". (2018). (accessed: August, 2018).

[6] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A congestion sensitive website fingerprinting defense. In *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 121–130.

[7] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A systematic approach to developing and evaluating website fingerprinting defenses. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 227–238.

[8] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: Website fingerprinting attacks and defenses. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 605–616.

[9] N. Carlini and D. Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57. https://doi.org/10.1109/SP.2017.49

[10] Heyning Cheng and Ron Avnur. 1998. Traffic analysis of SSL encrypted web browsing. *Project paper, University of Berkeley* (1998). Available at http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep networks learning by exponential linear units (ELUs). In *in the International Conference on Computer Vision (ICCV15))*.

[12] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 332–346.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[14] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security Symposium*. USENIX Association, 1–17.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[16] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier. In *ACM Workshop on Cloud Computing Security*. ACM, 31–42.

[17] Andrew Hintz. 2003. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies (PETs)*. Springer, 171–178.

[18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. 448–456.

[19] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A critical evaluation of website fingerprinting attacks. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 263–274.

[20] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 27–46.

[21] Simonyan Karen and Zisserman Andrew. 2015. Very deep convolutional networks for large-scale image recognition. (2015).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 1097–1105.

[23] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature* 4 (2015), 436–444.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. 86 (1998), 2278–2324. Issue 11.

[25] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. 2016. Systematic evaluation of CNN advances on the ImageNet. *CoRR* abs/1606.02228 (2016).

[26] Se Eun Oh, Saikrishna Sunkam, and Nicholas Hopper. 2018. *p*-FP: Extraction, Classification, and Prediction of Website Fingerprints with Deep Learning. "https://arxiv.org/abs/1711.03656.pdf". (2018). (accessed: August, 2018).

[27] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. Website fingerprinting at Internet scale. In *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 1–15.

[28] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 103–114.

[29] Mike Perry. 2013. A critique of website traffic fingerprinting attacks. Tor Project Blog. https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks. (2013). (accessed: December, 2015).

[30] Mike Perry. 2015. Padding Negotiation. Tor Protocol Specification Proposal. https://gitweb.torproject.org/torspec.git/tree/proposals/254-padding-negotiation.txt. (2015). (accessed: October 1, 2017).

[31] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2018. Automated Website Fingerprinting through Deep Learning. In *Proceedings of the 25nd Network and Distributed System Security Symposium (NDSS 2018)*. Internet Society.

[32] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote identification of encrypted video streams. In *USENIX Security Symposium*. USENIX Association, 1357–1374.

[33] V. Shmatikov and M. Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security (ESORIC)*. Springer, 18–33.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[35] Q Sun, DR R Simon, and YM M Wang. 2002. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 19–30.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. (June 2015).

[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11 (2010), 3371–3408.

[38] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security Symposium*. USENIX Association, 143–157.

[39] Tao Wang and Ian Goldberg. 2013. Improved website fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 201–212.

[40] Tao Wang and Ian Goldberg. 2016. On realistically attacking Tor with website fingerprinting. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter Open, 21–36.

[41] Tao Wang and Ian Goldberg. 2017. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *USENIX Security Symposium*. USENIX Association, 1375–1390.

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 3320–3328. http://dl.acm.org/citation.cfm?id=2969033.2969197

A DEEP FINGERPRINTING (DF) MODEL'S ARCHITECTURE FOR WF ATTACKS

One of the compelling properties for CNN is the transferability of the model. The transferability refers to the ability of the model to be used as a base model for similar tasks. Instead of training an entire CNN from scratch, the researcher can adapt the model to a similar task, specifically with a similar input format.

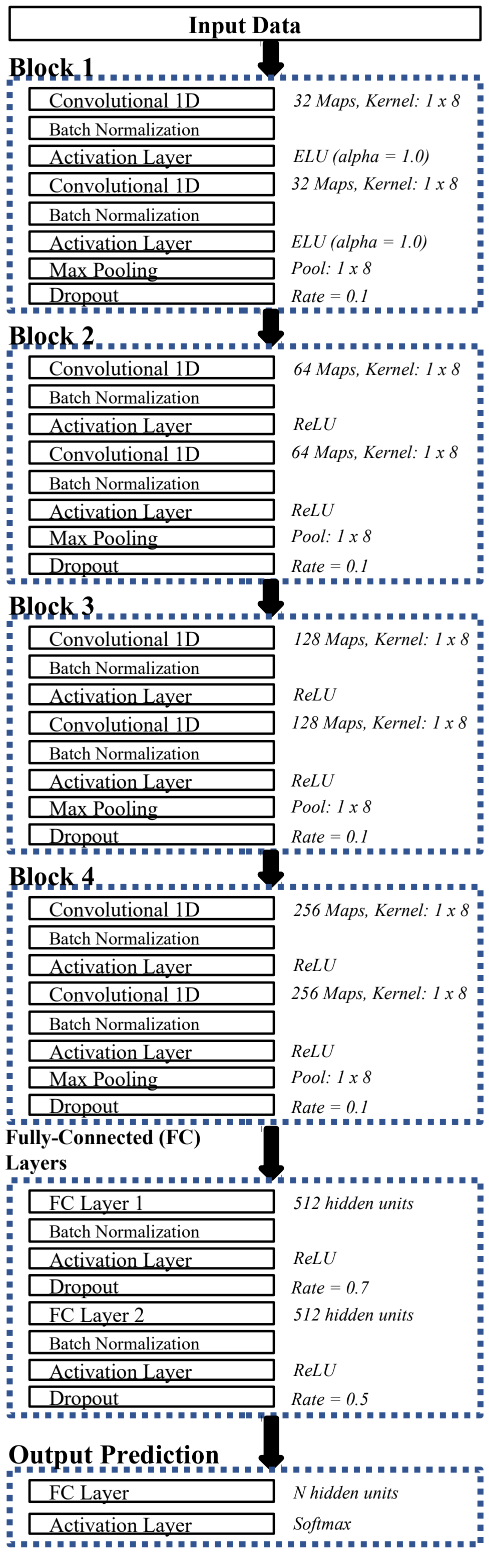


Figure 8: Our design of DF model's architecture used in WF attacks

In WF research, to the best of our knowledge, we are the first who provide the full technical details, guidelines and suggestions on how to implement our CNN-based DF model to perform WF attacks. In this section we provide details for our DF architecture and its hyperparameters to create our model, and to allow other researchers to apply it in their future work (see Figure 8):

Input Data. The input data for our DF model is the vector of packets' directions with length 5,000 (1 x 5,000). We initially tried adjusting the input dimension to be a matrix of shape similar to the matrices typically fed into CNNs for image recognition tasks (e.g., 50 x 100 pixels). The accuracy for 2D input was reasonably good, but slightly lower than 1D input. The major difference is training time: 1D input is significantly faster than 2D input, even though the total number of data points is the same for both input dimensions. We presume this difference results from tensor operations that have to deal with higher dimensions of data. We suggest that for the WF task, it is more appropriate to use 1D input as it is faster for training and provides better classification performance

Convolutional Layers (Block 1). Figure 8 describes the architecture of our DF model divided by blocks, where a block comprises a set of convolutional layers, a batch normalization layer, a max pooling layer and a dropout layer. The first block in the DF is specially important due to its proximity to the input.

As we mentioned in Section 5, since the nature of our input is different to inputs considered in image recognition, we had to find an activation function that fits our input values. We chose the Exponential Linear Unit (ELU) because prior work has shown that it provides fast and accurate classification with negative inputs [11, 25]. The results obtained from hyperparameters tuning suggested that applying ELU in the first two convolutional layers followed by ReLU with the rest of convolutional layers provides the best accuracy compared to only using ReLU. This suggests that ELU plays an important role in extracting hidden features from the input data.

Dropout Regularization. CNN-based models are specially vulnerable to overfitting, an issue that might be easily overlooked by the developer. We applied a dropout technique to mitigate overfitting in the design of our DF model. Our strategy to apply dropout was to embed in between feature extraction (Blocks 1-4) and classification (Fully-connected layers) using different rates. In feature extraction, we deployed dropout right after the max pooling layer in each block with 0.1 dropout rate. In addition, we added a dropout layer after each fully-connected layer with rate 0.7 and 0.5, respectively. As we observed from the hyperparameters tuning, the overfitting mostly arises at the fully-connected layers, and it is less problematic at the convolutional layers. Thus, we adjusted different dropout rates appropriately according to this observation.

Batch Normalization. We studied the technique to accelerate model learning called *Batch Normalization (BN)* [18]. This technique provides benefits to improve the classification performance in order to, for instance, learn faster while maintaining or even increasing accuracy. Moreover, it also partially serves as a regulation method as well. Thus, we applied BN and dropout regularization together and obtained a boost in both performance and generalization. However, adding BN layers requires additional training time. We observed that it added around 100% training time for each epoch compared to the model that did not apply BN. Yet, we believe it is worth applying BN, as the additional training time is compensated with a faster learning rate (it requires less number of epochs to reach the same level of accuracy) and can ultimately achieve higher testing accuracy. In our model, we applied BN right after every convolutional and fully-connected layers.

In conclusion, the researcher can apply this model and our suggestions to develop their own CNN-based model for WF. There are other details that we cannot describe here due to limit space including number of filters, kernel size, stride size and pool size. However, we will ensure that our implementation details, along with the source code and data used in this study, will be published on a website upon publication of this paper, so that researchers can reproduce our results.

B ATTACK PERFORMANCE METRICS

In this section, we define and justify the metrics we have used to evaluate the success of the attacks. There are two scenarios under which WF attacks are evaluated: *closed-world* and *open-world.*

B.1 Closed-world Evaluation

In the closed-world scenario, we assume that the user is limited to visiting a fixed set of websites, and the attacker knows this set and can train his classifier on it. In this scenario, the success of the attacker is simply measured as the ratio of the number of correctly classified traces to the total number of traces, we call this ratio the attack's *accuracy.*

$$Accuracy = \frac{P_{correct}}{N} \tag{1}$$

$P_{correct}$ is the total number of correct predictions. A correct prediction is defined as the output of the classifier matching the label of the website to which the test trace belongs. N is the total number of instances in the test set.

B.2 Open-world Evaluation

In the open-world scenario, the user may visit any of a large number of websites. Since the attacker cannot effectively train on so many sites, he selects a relatively small set to train his classifier on (the *monitored set*). For experimentation, we model the rest of the Web using a set of sites that the attacker does not try to identify with WF (the *unmonitored set*). Note that the unmonitored set is more than two orders of magnitude larger than the monitored set in our experiments. As mentioned in Section 5.7, we measure *Precision* and *Recall* in this scenario.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

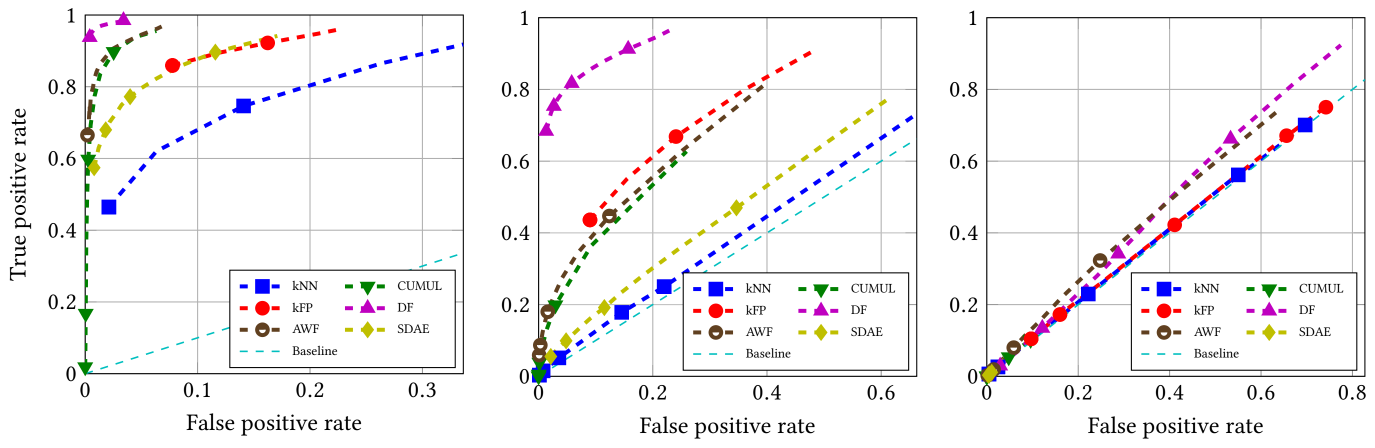
Where:

- TP is the total number of test samples of monitored websites that are correctly classified as monitored websites.
- TN is the total number of test samples of unmonitored websites that are correctly classified as unmonitored websites.
- FP is the total number of test samples of unmonitored websites that are misclassified as monitored websites.
- FN is the total number of monitored websites that are misclassified as unmonitored websites.

In addition, the attacker can measure *precision* and *recall* to tune the system. If his primary goal is to *reliably* determine that a user has visited a particular monitored website, one can try to decrease false positives at the cost of true positives and thus increase the precision of the attack. On the other hand, if the attacker aims to cast a wide net and identify *potential* visitors to the monitored web sites, then recall is more important, and the adversary should tune the system to increase true positives at the cost of additional false positives.

C OPEN-WORLD ROC CURVE

We plot the ROC curve for all the WF attacks against non-defended, WTF-PAD and W-T datasets using the standard model in the open-world scenario as shown in Figures 9a–9c. The ROC curve allows us to evaluate the classifier and strive for a trade-off between TPR and FPR. For example, the best overall results for DF against non-defended traffic might be optimizing for high TPR, with 0.98 TPR and 0.03 FPR, and optimizing for low FPR, with 0.94 TPR and 0.004 FPR.



(a) Non-defended dataset

(b) WTF-PAD dataset

(c) W-T dataset

Figure 9: ROC curve in Open-world scenario