Traffic Analysis with Deep Learning

Se Eun Oh
CSE department
University of Minnesota
Minneapolis, USA
seoh@umn.edu

Saikrishna Sunkam
CSE department
University of Minnesota
Minneapolis, USA
sunk0015@umn.edu

Nicholas Hopper
CSE department
University of Minnesota
Minneapolis, USA
hoppernj@umn.edu

***Abstract*—Deep Neural Networks (DNN) has obtained enormous attention with its advantageous feature learning and its powerful prediction ability. In this paper, we broadly study the applicability of deep learning to traffic analysis and present its effectiveness on the feature extraction for state-of-the-art machine learning algorithms, website and keyword fingerprinting attacks, and the prediction on the fingerprintability of websites. To the best of our knowledge, this is the first extensive work to introduce various applications using DNN in traffic analysis. With great help of DNN, the quality of cutting edge website fingerprinting attacks is upgraded while the feature dimension becomes much lower. As the classifiers, DNN successfully detects which website the user visited among 100 websites with 91% TPR and 1% FPR against 100,000 background websites, and as the fingerprintability predictors, it almost perfectly determines the fingerprintability of 4,500 website traffic instances with 99% of accuracy.**

I. Introduction

Tor is perhaps the most widely-used anonymity and censorship circumvention tool on the internet. Because of this widespread use (by over 2 million users per day), many academic papers have demonstrated that Tor is to some extent vulnerable to network-level traffic analysis attacks. A representative attack is website fingerprinting (WF) [32] that monitors and analyzes network traffic between a Tor entry guard and the client to identify the websites visited by the user, or to attempt to differentiate between visits to *monitored* (sites that might be targeted for censorship) or *unmonitored* sites.

Many security researchers have adopted a variety of machine learning algorithms for WF. Wang *et al.* [31] utilized k-Nearest Neighbors (k-NN) with a new weight learning technique, Panchenko *et al.* [21] proposed a scalable attack with Support Vector Machine (SVM), and Hayes and Danezis [12] adapted Random Forest classification (k-FP) to be resilient to WF defenses and noise. Classifiers introduced by prior works successfully reached high true positive rate (TPR) and low false positive rate (FPR) at the cost of a great deal of human effort in manual feature engineering. The primary contribution of each of these works was to improve the performance of classifiers by introducing new classifiers or discovering new feature sets.

Recently, Deep Neural Networks (DNN) have taken center stage in diverse research areas such as image recognition [9], [13], [27], game playing [30], and speech recognition [28], since they achieve flexibility in both learning complex functions with high precision and in no time-consuming feature engineering. DNN is an unprecedented model to support powerful automatic feature learning, which enables fitting any type of arbitrary complicated functions without prior knowledge, and learning different levels of abstractions of input data using multiple layers [6]. DNN is especially well suited to image analysis and interpretation and therefore, hospitals and laboratories have gained growing benefits by using DNN in medical image processing for diagnosis and treatment purposes [19].

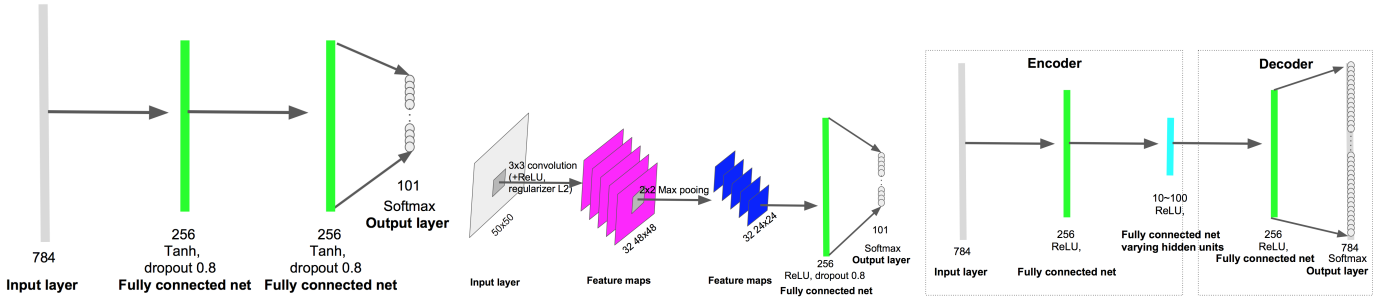
Apart from these widely-known applications, there have been several efforts to apply DNN to traffic analysis. Wang [34] adopted DNN for protocol detection such as SSL, Skype, and Gmail. Schuster *et al.* [26] applied DNN to encrypted video stream analysis to identify which streaming videos a user watched. Rimmer *et al.* [24] revisited WF with deep learning algorithms using an immense dataset. However, in the latter work, open world experiments, in which we assumed that the user visited websites out of the attacker's monitored list, were limited to binary classification (monitored vs non-monitored) and evaluation was restricted to Tor traces of websites.

In this paper, we extensively explore DNN from two different perspectives, as a feature extractor and a classifier. For feature extraction, we present how other machine learning algorithms, used in state-of-the-art WF attacks, perform with data abstraction learned by an unsupervised DNN. We show that the results of existing WF attacks with hand-tailored feature sets can be matched using the same classifiers and feature selection performed by an unsupservised autoencoder (AE) network.

As a classifier, we apply DNN to different levels of applications; we show that DNN can match the performance of WF attacks on Tor and TLS-encrypted traces, and keyword fingerprinting (KF) attacks over Tor. We also show that a DNN can be applied as a predictive defense mechanism, by showing that DNN can be trained to predict the fingerprintability of a website based on features extracted from its HTML source code. We demonstrate that DNN is well suited for both fingerprinting and fingerprintability analysis and performs well across multiple network traffic datasets.

Contributions. We summarize our key findings as follows:

Feature Engineering. We automatically extracted features using an autoencoder, fed them into k-NN, SVM, and k-FP



(a) Multilayer perceptron (b) Convolutional neural network. (c) Autoencoder

Fig. 1: Our architectures on 3 DNN models.

classifiers, and revisited state-of-the-art WF attacks to show how features extracted by an AE compare to hand-tailored feature sets in recent WF attacks. In addition, we varied the size of feature vectors learned by the AE, to determine its impact on the performance of these classifiers.

Even with extracted feature vectors of dimension 20, classification results of all machine learning techniques are comparable to or better than those of existing WF attacks. Interestingly, if we use Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN) as classifiers, the prediction is still as accurate as the best results of other WF attacks even without additional feature extraction steps. AE-based feature extraction gives the contribution to traffic analysis using state-of-the-art machine learning techniques by significantly removing the cost of manual efforts on feature engineering and running time of classification algorithms.

Application of DNN to various fingerprinting settings. We studied the suitability of DNN to website fingerprinting, using MLP and CNN classifiers, in a variety of settings. We evaluated both classifiers in diverse experimental settings, by 1) considering different applications: WF and KF, 2) using different types of traffic: SSL traffic, Tor traces after applying WF defenses, and time lag-based dataset, which reflects the time gap between instances of network traces in dataset, 3) restricting the decision of classifiers with confidence thresholds, and 4) using background datasets of varying sizes.

CNN is a better model for WF because decisions of classifiers are less affected by confidence thresholds. In contrast, MLP is better suited to KF since CNN fails to have consistency in classification decisions for different portions of the dataset. MLP works adequately under WF defenses, BuFLO [10], Tamaraw [8], and Walkie-Talkie [33], however it performs much better against Tamaraw than other WF attacks. MLP also successfully detects whether the user visited one of Alexa top 100 websites over TLS-encrypted traffic even without considering packet size information. WF achieves very strong performance; CNN architecture produces a multiclass classifier with 95% true positive rate and 0.04% false positive rate to identify 100 monitored websites against 100,000 unmonitored websites.

Predicting Fingerprintability We are the first to attempt to construct feature sets based on elements in HTML documents such as statistics on links and embedded web contents and adopt MLP to predict whether the website is fingerprintable or not. We labeled features with *accuracy* of a website, which is the proportion of correct predictions in the total number of testing instances of a website and used diverse fingerprintability thresholds (10-90%) to be compared to a website *accuracy*. With these strategies, the fingerprintability analysis is reduced to the problem of answering whether the *accuracy* of a website is greater than a certain threshold or not. To study the sensitivity of predictions to the mix of traffic used in training and classification, we computed the accuracy and error rate by applying different class weights.

We predicted the fingerprintability of 4,500 instances of Alexa top 50 websites with 98-99% accuracy and less than 0.02% mean squared error (MSE) under 10-40% and 90% fingerprintability thresholds. To ensure better prediction, we propose to set the fingerprintability threshold as low as or as high as possible. Decisions made by DNN classifiers are robust against increasing the size of the background dataset, that is, even with 100,000 unmonitored traffic traces, we are able to achieve almost the same accuracy and very low MSE. Moreover, MLP successfully detects the fingerprintability of unpopular Alexa 85 websites with 98% of accuracy and 1% of MSE. Our fingerprintability classifiers are powerful across various scales of background set and different testing dataset. In addition, based on classification results by k-FP classifiers, we labeled HTML features and predicted the fingerprintability of websites using DNN classifiers. We achieved similar results. These results suggest that website designers interested in helping users avoid WF attacks (or onion service designers interested in protecting the location of their servers) can apply our classifiers to the HTML source of their sites to predict the vulnerability of content pages and alter them accordingly.

II. Background

A. State-of-the-art attacks

The importance of feature engineering for WF with cutting edge machine learning algorithms was highlighted by Panchenko *et al.* [22]. Their hand-built features based on traffic volume and timing were powerful enough to have high TPR in an open world scenario. After this work, many researchers

have exerted great effort to improve the performance of classifiers by discovering new feature sets, enhancing previously used algorithms, and introducing new classification algorithms.

Wang and Goldberg [32] proposed Tor cell sequences as a new feature set and used SVM with distance based metrics to yield classifiers with 95% TPR and 0.2% FPR. The *k*-NN classifiers with revised weight learning, proposed by Wang *et al.* [31], enabled the attacker to achieve higher TPR than prior WF attacks.

Panchenko *et al.* [21] demonstrated more scalable WF with larger background datasets than previous research. They proposed using SVM with a new feature set based on cumulative traces and obtained 96-97% TPR, higher than *k*-NN classifiers [31]. Hayes and Danezis [12] conducted a thorough analysis on new categorical features and built the classification models based on Random Forest with Hamming distance. Their classifiers were less susceptible to padding-based WF defenses such as BuFLO [10] and Tamaraw [8].

Oh *et al.* [20] extended WF with new feature sets to launch a KF attack, which identifies keywords that the user typed. They used SVM with RESP feature sets (svmRESP) that focus on the response traffic to receive embedded web objects from search engine results. They achieved 83% TPR and 8% FPR with svmRESP to correctly identify 100 monitored keywords against 10k unmonitored keywords.

These works focused on finding feature sets and machine learning models, which were better suited to their datasets. Their methodologies always required a great deal of human efforts on feature analysis and time-consuming trial-and-error procedures to determine better classification algorithms.

In Section VI-A, we study the functionality of feature engineering of an autoencoder by revisiting more recent 3 WF attacks [12], [21], [31] with new features, extracted by an AE, to show how existing WF attacks perform. We also explore whether the variation in the size of AE features impacts the effectiveness of classifiers by changing the number of units in a hidden layer encoding features.

B. Deep neural network

In this section, we briefly discuss Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), which are methods for supervised learning, and Autoencoder (AE), which is an unsupervised learning method.

Multilayer Perceptron. MLP [11], [25], shown in Figure 1a, is a basic neural network, a sort of feed forward network, and also is known as a backpropagation algorithm. It consists of an input layer, one or more hidden layers and an output layer. Therefore, MLP always has at least 3 layers. In MLP, all nodes are fully connected between layers and its series of two procedures, *forward propagation*, which initializes weights and forwards pass through multiple layers to produce the output, *back propagation*, which calculates errors of the output layer, and then updates weights layer by layer. A single pass through is called an *epoch* and consists of multiple *batches*.

Convolutional Neural Network. CNN [17], shown in Figure 1b, consists of one or more convolutional layers, followed by one or more fully connected layers. The forward propagation runs three series of operations, *convolution*, *pooling*, and *classification*. The *convolution* operation extracts features from the input by learning features using small squares of input, which are called *filters* or *kernels*, and calculating dot products to generate a feature map. Interestingly, sliding different filters over the same feature generate different feature maps.

Pooling reduces the dimension of the feature maps and also extracts important elements from the feature maps. Thus, max pooling means selecting the max element from the feature map. Then, high level features learned by convolutional and pooling layers are fed into MLP for the *classification* and to learn non-linear relationships between features. Then, CNN proceeds *back propagation* in the same manner as MLP.

Autoencoder. AE [5], shown in Figure 1c, consists of two neural networks, an *encoder*, which learns lower-dimensional data abstractions, and a *decoder*, which recovers the original data. It provides benefits to reduce the feature dimensionality for data visualization and reducing the noise in the data. *Hidden units* in an encoder keep important information and at the same time remove noise. This eventually leads to a more efficient and meaningful data representation. In Section VI-A, we focus on this functionality of dimensionality reduction [14]. We used MLP for an *encoder* and a *decoder*, while varying the number of *hidden units* in a hidden layer of an encoder.

To construct a more generative model, Variational AE (VAE) was introduced by Kingma and Welling [16] and Rezende *et al.* [23]. Instead of memorizing a fuzzy data structure, it generates latent vectors following a Gaussian distribution by forcing a constraint to an encoder. Subsequently, to compute the loss of a VAE, two types of losses must be considered, the error between the input and reconstructed data, and the loss between latent variables and a unit Gaussian, reflected by KL divergence. Training VAE is tricky due to the trade off between these two different losses. Improvement in the generalization also promotes the quality of data reconstruction by a decoder.

Avoid overfitting. DNN usually struggles with overfitting, which means that the network memorized the training samples and hence, the error in testing dataset is large even though the training loss is tiny. To overcome this issue, early stopping, dropout [29] and regularization are widely used. Early stopping ends training when the loss does not decrease. The dropout temporarily removes units in layers based on the probability of each unit to be retained. Regularization is also known as weight decay, which means that it penalizes large weights based on constraints on their squared values (L2) or absolute values (L1). We used all three of these techniques to avoid overfitting for both MLP and CNN classifiers in Section VI.

In this paper, we used MLP and CNN as classifiers and an AE for feature compression. Implementation details in all three networks are introduced in Section IV-C and experimental results about how MLP and CNN perform for traffic analysis (e.g.,WF, KF, etc.) are shown in Section VI-B, VI-C, VI-D, and VI-E. The prediction accuracy of state-of-the-art traffic

TABLE I: Website fingerprinting with MLP classifiers using different size of feature dimension (using 20k background set)

Feature Size	Multiclass			Binary		
	TPR	FPR	BDR	TPR	FPR	BDR
784	94±1	5±1	89±1	95±1	0.3	99.2
1296	94±1	5±1	90±1	95±1	0.5	98.8
2500	94±1	5±1	89±2	94±1	0.8	98.2

analysis with AE-based features is presented in Section VI-A. Lastly, we also evaluate MLP and CNN for the prediction on the fingerprintability of websites in Section VI-F.

III. Feature Extraction

A. Dataset Overview

Website Tor traces (WTT). For monitored traces, we used Tor traces (**WTT-wang**), provided by Wang *et al.* [31] and we collected the **WTT-time** dataset consisting of 90 instances of each of Alexa top 100 websites [2]. This dataset additionally includes HTML document files for website traffic instances to be used in Section V and reflects the time gap, a week, between 4 different batches where each batch collects 110 traffic instances of each of 100 websites. Therefore, one batch started and the next batch executed after a week. We collected it from June to August in 2017.

For Section VI-F, we further collected 60 instances of 85 Alexa websites, ranked around one million. This dataset was collected in September and we used the same batch settings as in WTT-time dataset collection. For background traces, we used the dataset, provided by Hayes and Danezis [12], which consists of one instance of each of one million websites.

Keyword Tor traces (KTT). For both monitored and background traces, we used Google query traffic instances, provided by Oh *et al.* [20], which contain 100 instances of each of 100 top-ranked monitored keywords and one instance of each of 80,000 background keywords.

Website TLS-encrypted (non Tor) traces (WST). We harvested the website traces in normal web browser settings and the dataset consists of 90 instances of each of Alexa top 100 websites for a monitored set and 9,000 Alexa websites excluding monitored websites.

B. Features for DNN

Based on various traffic dataset, we investigated optimal features for MLP and CNN classifiers.

Tor traces for WF. Since Tor supports a specific transmission unit, known as a Tor cell (512 bytes), we extracted a Tor cell sequence, which consists of 1 and -1 indicating that the client sends 512 bytes or receives 512 bytes, respectively. The Tor cell trace is also similar representation of features used in hand-written digits [1], which consist of 1 or 0. In addition, since all instances should have the same feature dimension, we experimentally determined the optimal number of features, which made classifiers more effective.

Based on the distribution of Tor cell trace lengths in monitored dataset (Figure 13a in Appendix A), we experimentally selected three effective sizes, 784 features, 1296 features, which was a median , and 2500 features, which was a maximum. In addition, we padded 0 for instances whose number of features were lower than these cutoffs. Based on Table I, we chose 784 features, which was the lowest dimension and as well yielded competitive results, and used it for all WF experiments in Section VI.

Tor traces for KF. We evaluated two feature sets, Tor cell traces and RESP traces, provided by Oh *et al.* [20]. For RESP features, we did not use the cumulative setting because it gave us lower accuracy. Based on the size distribution of Tor cell traces in the keyword dataset (Figure 13b in Appendix A), we experimentally determined 2500 features yielding higher TPR as the optimal one and used it for all KF experiments in Section VI.

TLS-encrypted traces. We evaluated various feature sets, aggregated based on the inter-packet timing information, the length of tcp packet, and the length of tls record, and different feature dimensions to find the best feature setting, which yielded better results. Based on Table VIII, the timing feature was more informative to launch WF over TLS-encrypted traffic. However, this result conflicted with our expectation since encrypted traffic does not enforce a fixed size (e.g., no padding.) and thus, the packet size information should leak some traffic pattern for WF. Hence, to improve results for features, associated with B and C in Table VIII, we reconstructed those features with a -1 and 1 string tensor rather than using the number of tor cells sent or received. For example, for the trace [3,-2], we used [1,1,1,-1,-1]. This new representation gave us much higher results, 82% TPR. Experimental details will be discussed in Section VI-D.

C. Features with Autoencoder

While the autoencoder is widely used for reconstructing the data, it is also known for the ability of feature dimensionality reduction because an encoder projects the original feature vector into lower dimension representation. Since the performance of other machine learning algorithms is highly impacted by feature sets, features, extracted by an AE, are attractive inputs to those.

In this section, we explore how much features, compressed by an encoder, impact the performance of other machine learning algorithms, SVM, k-NN, and k-FP. In particular, these features can make SVM more efficient because its running time is linear to the number of features learned by the algorithm. Based on Figure 1c, to extract meaningful features by an autoencoder, we trained an encoder and a decoder and we captured feature vectors, learned and compressed by the second hidden layer in an encoder. We named them *encoded features*. We also varied the number of units, 10-100, in this hidden layer to compress the original features into various lower representations. We thoroughly evaluated other classification algorithms with them by comparing to state-of-the-art WF attacks in Section VI-A.

With the autoencoder, we additionally tested a variational autoencoder (VAE) and extracted encoded features as explained above, however, we failed to extract meaningful traces.

For VAE, we achieved 2 % TPR for multiclass classification and 3 % TPR for the binary decision. The reason for lower TPR with VAE was that we failed to reduce the loss, specifically KL divergence loss between latent space, which is an encoder's distribution and its probability density function. Since this gap was huge, KL divergence blew up and further led large cost. Since the VAE injects random Gaussian noise and optimizes the likelihood, the VAE performs more nicely for datasets where the latent is important [18]. Website traffic instances are less likely to have a reasonable estimated density that the log likelihood maximization returns than the image dataset.

IV. Traffic Analysis with DNN

To use deep neural network (DNN) for the traffic analysis, we adjusted DNN to our purpose. In this section, we introduce our DNN architectures and metrics to evaluate the performance of our classifiers.

A. Threat Model

We assume that there is a network-level, passive adversary who is only able to monitor and capture network traces, sent and received by users. Since the user uses a Tor browser, the adversary is only able to observe and harvest network traffic between the client and a Tor entry guard. In addition, there are no cooperative adversaries involved in this communication (e.g.,web servers, Tor relays, etc.).

Our attacker is interested in two classification problems, one is to determine whether a monitored trace is in his monitored list or not (binary classification), and the other is to correctly predict which website the user visited and which keyword the user typed (multiclass classification). Since the adversary carries out both WF and KF attacks, he uses different metrics according to fingerprinting tasks and details are introduced in Section IV-B.

B. Metrics

We discuss metrics and methodologies for appraising MLP and CNN classifiers.

Traditional metrics. To evaluate the performance of binary and multiclass classifiers in open world experiment settings, we used the following metrics, previously used in WF and KF works [12], [20], [31].

- True positive rate (TPR): This metric indicates the proportion of positive samples that are predicted as positive. That is, TPR represents how many monitored traces are correctly classified.
- False positive rate (FPR): This metric shows the proportion of negative samples that are mispredicted as positive. That is, FPR reflects how many unmonitored traces are misclassified as monitored traces.
- Bayesian detection rate (BDR): To reflect practicality in measuring the performance of our attack, we used BDR, computed by $P(M|V) = \frac{P(V|M)P(M)}{P(V|M)P(M)+P(V|B)P(B)}$, where V means that the user is visiting one of monitored websites, $P(M)$ is the probability that the website is included in monitored set and $P(B)$ is $P(\neg M)$. $P(V|M)$ is computed by TPR and $P(V|B)$ is estimated by FPR. This metric presents how much our attack is practically viable.
- Within-monitored accuracy (WMacc): In KF, for a fair comparison to svmRESP [20], we used a Within-monitored accuracy, proposed by Oh *et al.* [20] to measure the performance of multiclass classifiers. This is computed by the number of TPs, divided by the total number of monitored samples.

Confidence threshold. In DNN, the output layer returns the probability vectors consisting of prediction probabilities for all labels. Even though the label having the highest probability is selected as a predicted label, if its prediction probability is not high enough, the decision made by classifiers will not be reliable. Therefore, we additionally used the confidence thresholds, 10-90%, to generate more confident predicted labels. If the highest probability is less than the threshold, we regard this case as being detected as "others". Since confidence thresholds significantly impact the number of FNs and TNs, we exhaustively inspected how the performance of MLP and CNN classifiers is affected by increasing the confidence threshold and what the optimal threshold is to yield a high TPR as well as a low FPR.

TopK analysis. Based on the prediction probability vector, returned by the output layer, we are able to be given to other meaningful labels apart from the label with the highest probability if their probabilities are high enough to be trustworthy. We varied k, 1-5, and if there is a match between an actual label and the one in the top k list, and its probability is larger than the confidence threshold, we treat it as a TP.

In the open world experiment, if the negative (background) label is included in the top k list and the actual label is positive (monitored), we always treat it as a false negative even if the list includes an actual label. We used top k analysis in KF attacks(Section VI-C) and evaluation under WF defenses (Section VI-E), and this approach improved the performance of classifiers in both experiments.

C. Networks

We discuss how we built the following three DNN architectures for our attack.

MLP. Our MLP (Figure 1a) consists of 3 fully connected layers, where the first two layers contain 256 nodes and the number of units in the output layer corresponds to the number of classes in the dataset, which is 100 for closed world evaluations, 101 for open world experiments, and 50 or 85 for the fingerprintability analysis.

We used L2 regularizations for the first two hidden layers and the dropout between those layers to minimize the impact of overfitting on the validation of testing dataset. For the dropout, we used 0.8 for the probability of each unit to be retained in the layer.

For activation functions, we applied the tanh function to the first and second hidden layers and the softmax function to the output layer. We used Stochastic Gradient Descent (SGD) as

the optimizer to ensure faster backpropagation. To measure the error rate of classifiers, we used the cross entropy.

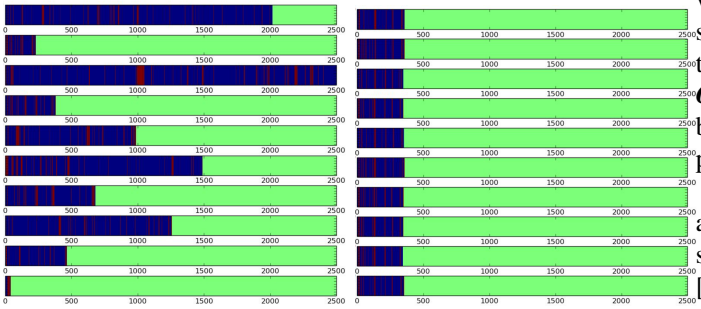
CNN. Our CNN (Figure 1b) comprises a convolutional 2D layer with 32 filters, followed by a 2D max pooling and two fully connected layers. The first fully connected layer consists of 256 units and the output layer contains the same number of units as which we used in MLP. In addition, we applied L2 regularization to the convolutional layer.

For activation functions, we used a rectified linear unit (ReLU) in the convolutional layer and the first fully connected layer, and the softmax function in the output layer. Similar to MLP, we used Adaptive Moment Estimation (Adam) for the optimizer to reduce the running time and applied the cross entropy to compute the loss in classification results.

AE. We extracted meaningful features by fetching lower abstractions, compressed by an encoder network and ran other machine learning algorithms with those new features to compare to recent WF attacks. Our autoencoder (Figure 1c) consists of an encoder and a decoder network and each consists of two fully connected layers. The number of units retained by the first and third hidden layer is 256 and for the output layer, we used the same number of nodes as the dimension of the original input data.

We changed the size of units in the second hidden layer to get different feature dimensions in compressed inputs to be fed into SVM, k-NN, and k-FP classifiers. To capture *encoded features*, we reused the session to get weights learned by an encoder and generated compressed vectors based on weights, saved in the session. In addition, we used the ReLU activation function, the Adam optimization, and mean squared error to calculate the loss of the classifier decisions.

Figure 2 shows the intuition behind why we chose MLP and CNN for our classifiers. Figure 2a presents 1D images of 10 Alexa websites and Figure 2b shows 1D images of 10 traffic instances of the same website. The blue color indicates -1 (incoming transmission), the red color indicates 1 (outgoing transmission), and the green color indicates 0, which is padded to make all sizes of traffic instances same. Based on Figure 2, website traffic instances are different each other enough to be differentiated by MLP and CNN classifiers.



(a) Traces of 10 alexa websites†. (b) Traces of 10 traffic instances of www.google.de.

Fig. 2: Strip images (1x2500) of website traffic instances

†www.zillow.com, www.foodnetwork.com, www.mail.ru, www.coursera.org, www.adcash.com, www.adobe.com, www.dailymotion.com, www.themeforest.net, www.livedoor.jp, and www.alipay.com

V. Analysis on HTML documents

According to the fingerprintability analysis explored by Oh *et al.* [20], the fingerprintability of traffic instances is affected by embedded web dynamic contents. Rather than doing high level analysis on factors to exert an effect on the fingerprintability, we downloaded 9,000 HTML document files of Alexa top 100 websites, observed elements of documents that impact the fingerprintability, and tried to construct features based on them.

A. HTML DOM features

Links and domains. First, we fetched all links in HTML documents and generated features based on the number of links and domains in an HTML DOM. In particular, we inspected how many links were from third party websites because we expected that downloading web objects from different web servers made the traffic pattern different with the case when all downloads occurred from a single web server.

Tag path. We generated all tag paths by constructing each tag path by adding a tag if it was nested. For example, for a document consisting of <html><body><a></body></html>, there are 4 tag paths, <html> (depth=1), <html><body> (depth=2), <html><body><a> (depth=3), and <html><body> (depth=3). Based on the list of tag paths, we are able to create statistics about the number of tags contained in a tag path, the frequency of increase or decrease in the size of tag paths, and the depth of tag paths. These features enabled us to measure the degree of the complexity or simplicity in web contents, embedded in web pages.

Tags and other elements. We computed statistics about the number of tags, attributes, and comments, and the number of characters and words in `data` and `style` attributes. We assumed that these features impacted the size of HTML DOM and consequently, the variance in these features affected the traffic pattern.

Embedded files. Diverse types of files are embedded in an HTML document file and fetching those resources have been strong traffic pattern for WF attacks. Thus, based on the finding that all image and video contents are nested in an `img` tag, we computed the number and the proportion of image and video files, contained in `img` tags. Furthermore, we located specific file extensions to obtain counts and proportions of them (e.g., jpg, gif, ico, html, etc.).

Other features. We calculated the total transmission time based on the start and end time in a traffic capture file (e.g., pcap), and the size of capture files and HTML document files.

Eventually, we created 65 features, named *HTML features*, and for each feature, we generated the rank for each instance. For example, if we have 3 instances, [3,19,...,10], [7,10,...,201], and [17,7,...,25], we converted features into the rank information, [1,3,...,1], [2,2,...,3], and [3,1,...,2] to be fed into classifiers to determine whether a website is fingerprintable or not. Note that based on this approach, our fingerprintability might not be globally applicable, however, we are able to overcome this by training classifiers with large dataset. We listed 65 features in Appendix B.

TABLE II: Number of instances of less than and greater than a FP threshold in 9,000 web trace instances.

FP thr	# less	# greater
10	68	8932
20	316	8684
30	680	8320
40	972	8028
50	2161	6839
60	3399	5601
70	5222	3778
80	6787	2213
90	8036	964

B. Classifiers for the fingerprintability.

Fingerprintability (FP) thresholds. After training and testing MLP classifiers with 9,000 monitored website traffic instances and 20,000-100,000 background instances, we computed the proportion of correct predictions for each monitored website to be used as *accuracy* of each website. Rather than restricting *accuracy* threshold to a certain cutoff, we applied 9 different thresholds, 10-90%, to determine if it is fingerprintable. For example, if the threshold is 10%, classifiers decide whether *accuracy* of the website is less than 10% or not. If yes, it is not fingerprintable.

Metrics. We used 4,500 instances of 50 websites for training dataset and 4,500 instances of other 50 websites for testing dataset. All *HTML features* of those website traffic instances were labeled with 1 or 0, where it was 1 if its *accuracy* was greater than a fingerprintability threshold, otherwise, it was 0. As expected, with this labeling, we were given to unbalanced number of instances for each class, as shown in Table II. To overcome the imbalance in the training dataset, we computed a total accuracy and a mean squared error by applying class weights, derived based on the size of training samples for different FP thresholds (Table II).

Gini importance. To measure the feature importance score for each of 65 features, we used Gini importance [7], which is derived as the total number of splits including target feature over the total number of samples that the feature splits, by running Random forest classifiers. In other words, this score means an average purity, earned by splits of target feature. We used Gini importance to see which *HTML features* are more important to determine the fingerprintability of websites in Section VI-F.

With FP thresholds and metrics, measured by considering class weights, we adopted our MLP classifiers to evaluate their applicability to the fingerprintability prediction. All experimental results are discussed in Section VI-F. The fingerprintability prediction with MLP classifiers is meaningful to suggest WF defenses with DNN models. Upon a request security checks for the website by web developers, FP classifiers can predict if the website design is vulnerable to the traffic analysis. We will leave further developing a defense tool based on FP classifiers as future works.

TABLE III: Performance of state-of-the-art machine learning algorithms with AE features. Note that dim indicates the dimension of features.

dim	Binary				Multiclass			
	10		80		20		80	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
k-NN	96±1	3±1	98±1	2±1	82±2	2±1	84±2	2±1
SVM	95	3	98	1	97	12	97	10
k-FP	94±1	2±1	96±1	1±1	81±2	1±1	83±1	1±1

VI. Experiment

In this section, we presented the effectiveness of DNN models for the traffic analysis with the following research questions. We named both MLP and CNN classifiers as DNN classifiers and models.

- Is an AE appropriate as the feature extractor for other machine learning techniques? (Section VI-A)
- How does the variation in experimental settings for DNN classifiers (e.g., network, confidence thresholds, etc.) impact the performance of WF attacks ? (Section VI-B)
- How much does the top k analysis affect classification results by DNN models? (Section VI-C)
- What is the optimal feature set to carry out robust WF attacks over TLS-encrypted traffic ? (Section VI-D)
- How much is DNN classifiers robust under recent WF defenses? (Section VI-E)
- Can DNN classifiers predict whether a website is vulnerable to WF attacks or not? (Section VI-F)

Experiment setup. We used Tensorflow [3] with TFLearn [4] front end for the implementation of DNN classifiers. We split the WTT, KTT, and WST datasets into training and testing dataset by the ratio, 9:1, and constructed different 10 folds, where each fold consists of randomly chosen monitored and background instances. We randomly selected the same number of instance indices for each monitored website to ensure that the number of classes in each fold was equal. We applied this 10 fold approach to all experiments done in this section.

We used 8 cores and 32GB of memory for the experiments using MLP classifiers and other machine learning algorithms, and 32 cores and 258GB of memory for CNN classifiers. With those resources, the longest job took 3 days to finish its 10 fold experiments. With a GPU, this running time would be considerably reduced.

As thresholds for early stopping, we used 500 epochs for MLP models and 300 epochs for CNN models because we observed that beyond those numbers, the testing accuracy decreased due to overfitting.

A. Recent WF Attacks with AE Features

As mentioned in Section IV-C, we revisited cutting edge WF attacks [12], [21], [31] and ran machine learning algorithms, chosen in their works, with new features, learned by an AE (AE features). Feature engineering, suggested by recent WF research [12], [21], usually requires negligible amount of human efforts such as manually inspecting the traffic pattern and

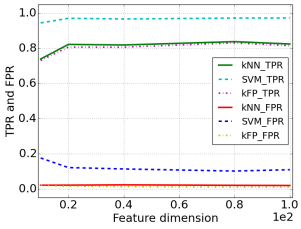


Fig. 3: The performance with various feature dimensions

Metrics	TPR	FPR
AE(80)+k-NN	97.89	2.07
AE(100)+SVM	97.6	1.47
AE(80)+k-FP	96.2	0.98
MLP	96.56	2.78
CNN	96.72	1.78

TABLE IV: The best performance using AE features based on WTT-wang dataset. Note that AE(n) indicates n features.

experimentally deciding the optimal size of feature dimension. The feature extraction based on an AE is effective to overcome this issue. Since the implementation of k-NN [31] is well suited to features based on a Tor cell trace, we implemented k-NN using `sklearn.neighbors` with different weight learning by considering the inverse of the distance between neighbors to handle AE features in a better manner.

In this experiment, the attacker tries to decide whether the user was browsing one of monitored websites or not (binary classification) and which website the user visited (multiclass classification). For all comparisons in Table III, we used 90 instances of each of 100 monitored websites and 9,000 background websites in WTT-wang dataset.

To understand how the size of AE features impacts the performance of classifiers, we varied the number of units, 10-100, in the hidden layer, which encoded website traffic vectors. In our architecture, this layer corresponds to the blue layer in Figure 1c. Since the dimension of this hidden representation is significantly lower than the length of the original input vector, this gives us the effect of dimensionality reduction. After we performed the open world evaluation based on WTT-wang dataset [31], we realized that the dimension of hidden units of an AE scarcely impacted the performance of machine learning algorithms in binary classification. Similarly, in multiclass classification, there was almost no change on both metrics for features whose dimension is larger than 20 (Figure 3).

As shown in Table III, AE features enabled classifiers to yield good performance as much as state-of-the-art WF attacks [12], [21], [31] presented, even with much lower feature dimensionality and no human efforts on the feature craft. We achieved much lower FPR than k-NN [31] and by the comparison to CUMUL [21] and k-FP [12], we obtained similar results in much shorter running time without expensive manual investigation in feature engineering. In particular, the benefit of faster running time had more paid off for SVM since AE features whose dimension is 20 took 39 minutes while CUMUL features took 4 hours. Running MLP and CNN classifiers led the comparable performance, as shown in Table IV, without separate feature extraction procedure. Based on these experiments, DNN's ability on feature extraction significantly lessens labours in the manual feature craft for other classification algorithms as well as generates cost-effective features.

TABLE V: WF with MLP using different size of background sets (TPR, FPR, and BDR (%)

Back Size	Multiclass			Binary		
	TPR	FPR	BDR	TPR	FPR	BDR
20k	94±1	5±1	89±1	95±1	0.3	99.2
50k	92±1	2	88±1	93±1	0.1	99.16
100k	90±1	1	87±1	91±1	0.07	99.10

TABLE VI: WF with CNN using different size of background sets (TPR, FPR, and BDR (%)

Back Size	Multiclass			Binary		
	TPR	FPR	BDR	TPR	FPR	BDR
20k	97±1	6±1	88±1	98±1	0.3	99.26
50k	95±1	3±1	85±2	96±1	0.1	99.14
100k	94±2	2±1	84±2	94±2	0.04	99.46

B. *WF Attack on Tor Traffic.*

We evaluated both MLP and CNN classifiers with WTT-time dataset in open world setting by varying the size of background sets and confidence threshold to show how much those factors affect the performance of classifiers.

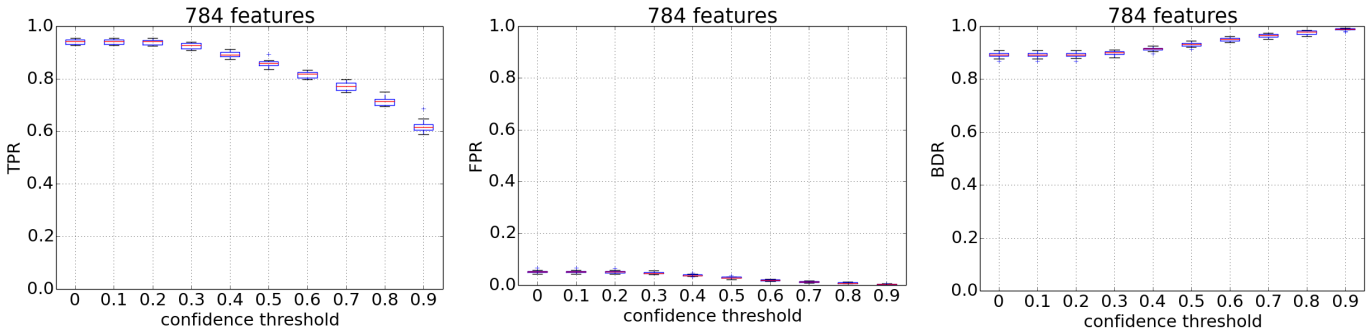
Background set. To show the effect of background set, we trained MLP and CNN classifiers with 101 labeled WTT dataset, where we have 9,000 traffic instances of 100 monitored websites and 1 traffic instance of 20,000, 50,000, and 100,000 background websites.

Increasing the size of unmonitored traffic instances weakened the performance of both classifiers but not significantly, since we achieved 90% TPR and 1% FPR for MLP multiclass classifiers even against 100,000 background dataset and the feasibility of WF attacks based on DNN models was presented with 87-89% of BDRs (Table VI). In particular, MLP and CNN classifiers made the attack, to detect if the user visited one of monitored websites, more promising with very low FPRs and very high BDRs based on Table V and VI.

Confidence threshold. To reflect the reliability of decisions made by MLP and CNN classifiers, we additionally applied 0-90% thresholds (0 corresponds to argmax) to prediction probabilities, returned by the output layer, to decide whether it is a confident TP or not, even though there is a match between an actual label and a predicted label with the highest probability. After we ran MLP and CNN classifiers with 9,000 monitored traces and 20,000 background traces, increasing the confidence threshold reduced the number of confident TPs, which lowered TPR, increased the number of TNs, which decreased FPR, and elevated BDR, which makes attacks more viable (Figure 4 and 5).

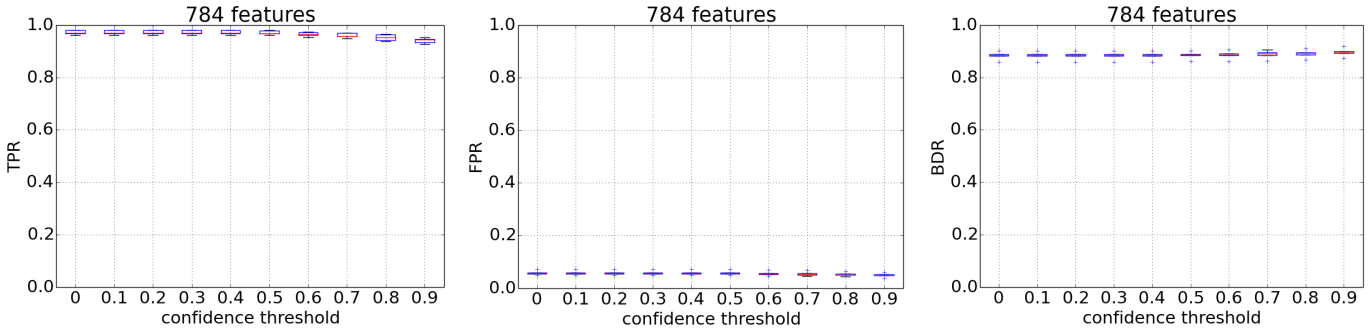
Network. Based on Table V and VI, CNN classifiers showed better results for all of metrics, except FPR in multiclass classification, however the gap between FPRs of both classifiers was not high. More interestingly, as shown in Figure 5, the increase in the confidence threshold hardly impacted the performance of classifiers across all metrics.

In summary, feeding more background traffic instances somewhat aggravates the performance of MLP and CNN classifiers, however, CNN classifiers still successfully determine



(a) True Positive Rate (b) False Positive Rate (c) Bayesian Detection Rate

Fig. 4: WF evaluation using MLP and different confidence threshold.



(a) True Positive Rate (b) False Positive Rate (c) Bayesian Detection Rate

Fig. 5: WF evaluation using CNN and different confidence threshold.

TABLE VII: KF with MLP(M), and CNN(C) using both RESP and cell traces. ((b):binary classification, (m):multiclass classification)

Metrics	RESP(C)	RESP(M)	Cell(M)
TPR(b)	86±1	88±1	59±2
FPR(b)	5	5±1	11±2
BDR(b)	95	95±1	85±2
WMacc(m)	22±1	27±1	22±1
BDR(m)	59±2	63±1	50±1

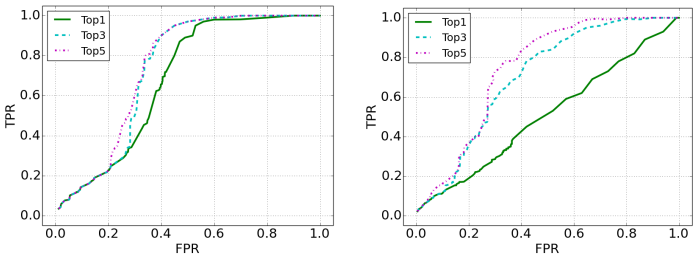
which website the user visited among 100 websites with 92% TPR and 1% FPR against 100,000 background website traces.

In particular, for the binary classification, both classifiers presented almost 0% FPR, which indicates that WF attacks based on MLP and CNN classifiers are feasible since it reduces the chances of misclassification and therefore, prevents blocking innocent users.

CNN classifiers will be a better choice for WF attacks because increasing the confidence threshold up to 90% led a little change in both TPR and FPR of CNN, which makes decisions more confident.

C. KF Attack on Keyword Dataset

We used 100 instances of each of 100 monitored keywords and 10,000 background keyword traffic instances in the KTT dataset and two feature sets, RESP traces and Tor cell traces, provided by Oh *et al.* [20].



(a) Resp trace (b) Cell trace

Fig. 6: ROC of Top-K analysis on KF for multiclass classification

Features. According to Oh *et al.* [20], RESP-based features make KF attacks more accurate. We explored whether it is consistent with MLP classifiers. Compared to Tor cell traces, Table VII shows that RESP features substantially enhanced the performance of MLP classifiers with much higher TPR and lower FPR. In particular, even with the comparison to results of svmResp in the binary classification, reported by Oh *et al.* [20], which were 82.6% TPR and 8.1% FPR, MLP

Metrics	A	B	C	AB	AC	BC	ABC
TPR	0.83	0.48	0.50	0.43	0.58	0.43	0.55
FPR	0.10	0.11	0.14	0.09	0.14	0.09	0.13

TABLE VIII: TPR and FPR of feature sets using WST dataset. A - inter-packet timing, B - TCP packet size, C - TLS record length

Metrics	packet time	TCP packet	TLS record
TPR (%)	83.03	81.74	82
FPR (%)	9.55	10.64	12.53

TABLE IX: TPR and FPR of inter-packet timing, -1/1 sequence of TCP packet size, and -1/1 sequence of TLS record length. Note that we used 10,000 features for -1/1 representation.

classifiers ensured better performance with 88% TPR and 5% FPR.

Top K analysis. We used WMacc in multiclass classification evaluation, as used in Oh *et al.* [20]'s work. We realized that the multiclass classification with MLP and CNN classifiers did not ensure good results as much as in binary classification, according to Table VII. To improve the performance of multiclass classifiers, we adopted the top k analysis, which considers additional predicted labels that also have reasonably high prediction probabilities. For instance, we selected the top k labels, which had higher probabilities than a certain threshold, and if the actual label was in those labels, we regarded it as a TP.

Based on Figure 6, the top k analysis helps to substantially improve the performance of multiclass classifiers for both Tor cell and RESP features. By using the top 5 analysis, MLP classifiers derived higher WMacc, which was 62%, than svmRESP classifiers [20], which was 55%.

Confidence threshold and network. Changing the confidence threshold impacts the performance of both binary and multiclass classification using MLP classifiers. The degree of distinction between metrics (e.g., TPR, WMacc, etc.) of RESP and Tor cell features became lower with a greater confidence threshold.

Overall, CNN classifiers were not an appropriate model for KF attacks for the following reasons. First, a greater confidence threshold leads larger standard deviation in all metrics, which indicates that decisions made by CNN highly depend on inputs, which are subsets randomly chosen in different folds. Thus, their prediction results are considerably variable according to different portions of dataset. Second, the performance of both classifications is not as good as MLP classifiers'. This is due to the dataset. Filters in CNN produce strong representations step by step based on the local input pattern in the dataset, which creates small pieces of the input and then assembles them in larger pieces. This is a highlighted feature of CNN, which significantly depends on local patterns in the dataset, which can be observed in images. In contrast, keyword traces are too similar to have such local dependence, which makes the performance of CNN poor.

In summary, MLP classifiers are a better choice for KF attacks. In the binary classification, they are more effective than state-of-the-art KF attack [20] and in the multiclass classification, they yield better results with the help of the top k analysis.

D. WF Attack on TLS-encrypted Traffic

In this section we evaluated MLP classifiers with TLS-encrypted traces. In particular, we investigated feature sets, which achieved better results, by varying feature representations and dimensions. We used WST dataset, consisting of 9000 monitored instances from Alexa top 100 websites (90 instances each) and 9000 background website instances.

Features. We built three types of sequences based on inter-packet timing (A), the size of TCP packet (B), and the length of TLS record (C), as shown in Table VIII. We obtained the highest TPR, 83%, with inter-packet timing sequences while aggregating features did not lead higher accuracy. Lower TPR with features based on packet size and record length was surprising since TLS-encrypted traffic does not use padding and this should let those traces have stronger traffic patterns.

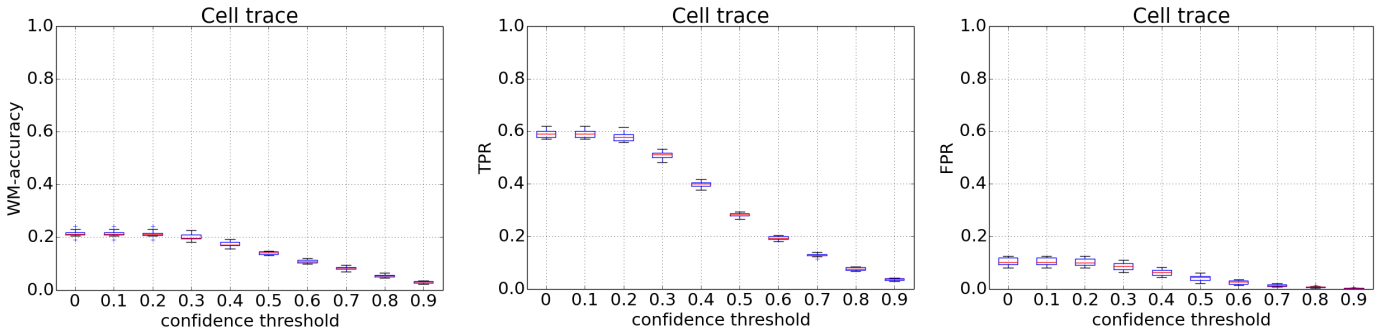
To improve those results, we converted them into the array consisting of -1 and 1, and the resulting vector then had the length equal to the absolute value sum of the original vector. We used 10,000 for the feature dimension, and according to Table IX, this modification significantly improved our TPR from 48% to 82% for TCP packet size, and from 50% to 82% for TLS record length.

Feature Dimension. With inter-packet timing sequences, we tried to tune the hyperparameter of feature dimension. We explored different feature dimensions from the average of traces, which is 660, to the maximum, which is 10,000. Based on the observation that taking the dimension size larger to incorporate more of each trace did not deliver higher accuracy as shown in the Figure 14 of Appendix C, we decided the optimal dimension size yielding higher TPR to be 1,000.

Based on the finding that -1/1 sequence led much better results, MLP classifiers performed better with the feature sequence consisting of binary information. This transformation is consistent with the intuition behind the batch normalization [15], which leads higher learning rates without sacrificing the accuracy by solving internal covariate shifts. With inputs based on -1 and 1, MLP classifiers can have less chances of the distraction to gradients (e.g., outliers), which is more likely to happen in original sizes of TCP packet and TLS record, and speed up the learning time.

E. Defense Traffic Analysis with DNN

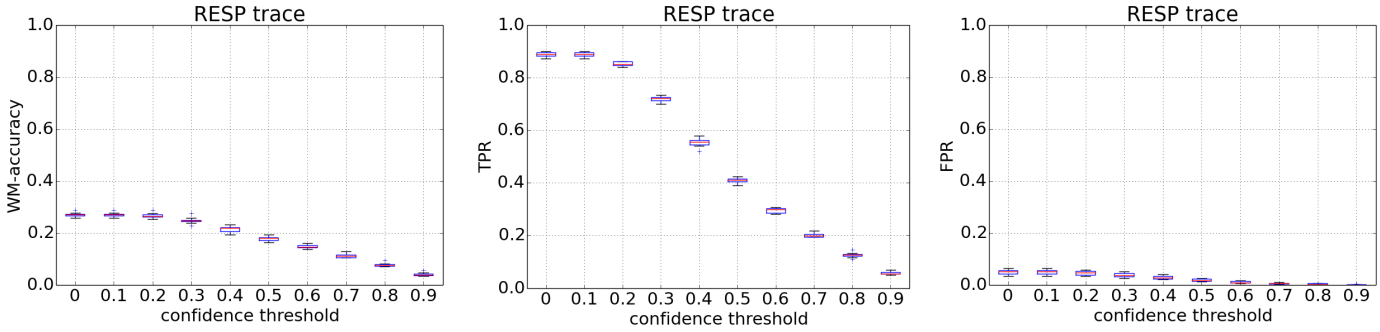
In this section, we evaluated MLP classifiers under recent WF defense mechanisms, BuFLO [10], Tamaraw [8], and Walkie-Talkie [33]. BuFLO [10] pads dummy packets to fill in the gap and further extends the transmission to send packets of fixed length at fixed intervals. Tamaraw [8] improves BuFLO to be a more efficient and effective defense by using different padding intervals for incoming and outgoing packet direction and fixing outgoing packets at a higher packet interval, which



(a) Multiclass Classification

(b) Binary Classification

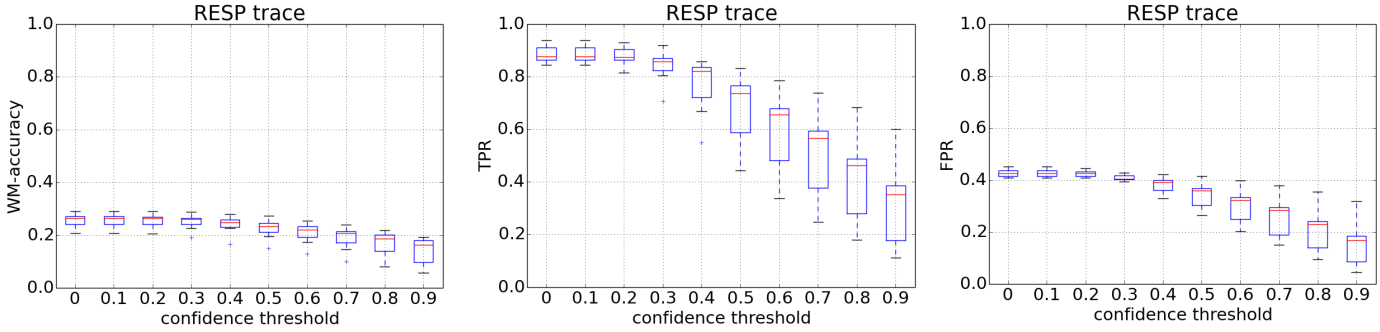
Fig. 7: KF(Cell) with MLP



(a) Multiclass classification

(b) Binary Classification

Fig. 8: KF(RESP) with MLP

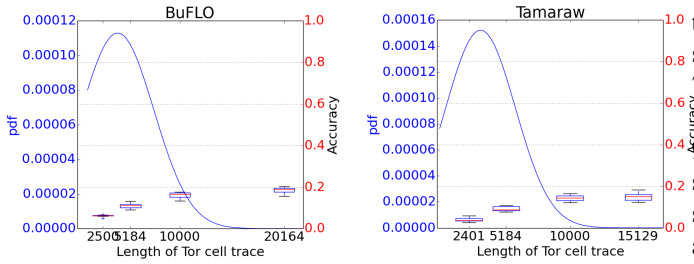


(a) Multiclass classification

(b) Binary classification

Fig. 9: KF(RESP) with CNN

reduces the overhead for infrequent outgoing traffic. Walkie-Talkie [33] is an efficient WF defense technique based on half-duplex communication and burst molding, which make packet sequences same and add fake cells to burst sequences to be molded into the supersequence. Due to the bandwidth overhead based on Figure 10, for BuFLO and Tamaraw, we used much larger feature dimension, 20,164 and 15,129, respectively. For Table X, we evaluated MLP classifiers using WTT-wang dataset in the closed world setting, which was similar analysis to the defense analysis by other WF research [12]. Based on Table X and Table 1 reported by Hayes and Danezis [12], MLP classifiers performed effectively as much as k-FP classifiers did under BuFLO and much better than other WF attacks under Tamaraw. In Table XI, we adopted the dataset, provided by Wang and Goldberg [33]. Based on Table 1 [33] and Table XI, even though WF defense sacrificed the accuracy of MLP classifiers, MLP classifiers still worked adequately and the top k approach somewhat improved the



(a) BuFLO (b) Tamaraw

Fig. 10: The accuracy of MLP increases with increasing the feature dimension

TABLE X: MLP performance under WF defenses.

Metrics	No defense	BuFLO	Tamaraw
argmax	0.86	**0.19**	**0.16**
Top1	0.86±0.02	**0.19**±0.01	**0.15**±0.02
Top3	0.91±0.02	**0.35**±0.02	**0.26**±0.02
Top5	0.91±0.02	**0.38**±0.03	**0.27**±0.03

performance of MLP classifiers.

F. Investigation on Fingerprintability (FP) using DNN

To construct training and testing data, fed into FP classifiers, we need to achieve *accuracy* of each website (e.g., the total number of TPs divided by the total number of instances of the website) to be used to label feature with 1 or 0, as discussed in Section V-B. To obtain *accuracy* of training websites, we trained and tested MLP classifiers with 90 instances of Alexa top 50 monitored websites (WTT-time dataset) and 9,000 background website traffic instances, and computed *accuracy* of each website. We took same procedures with Alexa top 61-100 monitored websites to achieve *accuracy* of the testing websites.

As discussed in Section V, based on HTML document files, we constructed 65 *HTML features*, shown in Appendix B, computed ranks for each feature, and labeled instances with 1 or 0, which means that the corresponding website *accuracy* is greater than a certain FP threshold (In other words, the website is *fingerprintable*) or not, respectively. Then, we used MLP classifiers as FP classifiers with those features and labels.

FP threshold. To define the fingerprintability of website traffic instances, we varied *accuracy* thresholds, 10-90%, to show how increasing the threshold impacts the performance of classifiers.

TABLE XI: Performance comparison between WF attacks under Walkie-Talkie. For Top k analysis, we chose the confidence threshold, which gave us optimal TPR. Note that undefended means traces, collected without the defense and defended indicates traces, collected under the defense.

Top$k$	Undefended	Defended
Top1	**0.84**±0.05	**0.26**±0.03
Top3	**0.88**±0.03	**0.37**±0.04
Top5	**0.89**±0.03	**0.38**±0.03

As shown in Figure 11a, if we increased the FP threshold, the accuracy of classifiers continually decreased and the mean squared error (MSE) increased until the threshold reached 70%. After that, the accuracy was increasing with decreasing MSE. According to Figure 11, we achieved the accuracy of 98-99% for 10-40% thresholds and 95% for 90% threshold. This shows that the FP classifiers performed better with relatively smaller or higher thresholds. Section V-B described how we applied class weights to derive both the accuracy and the error rate.

Feature importance. We adopted Gini importance, explained in Section V-B, to derive the feature importance score and Figure 12 shows those scores for different confidence thresholds. Then, we computed the sum of those scores for each of 65 features to select the top 15 features, shown in Table XII.

The total number of unique domains (feature index 5) and third party links (feature index 3) are more important *HTML features* based on Alexa top 100 websites and this is consistent with our expectation that the traffic regarding the communication with third party websites impacts traffic pattern.

In addition, the number of characters and words and embedded web objects such as images (feature index 30,40,50-55), included in HTML document files, are also important features. This indicates that simpler web pages carrying less amount of embedded web contents (e.g., texts, images,etc.) are more fingerprintable than complicated web pages.

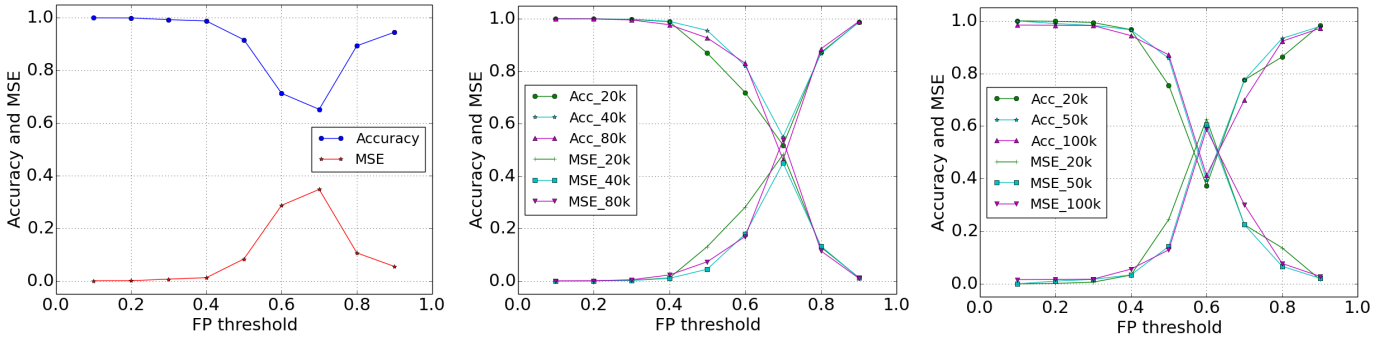
All these top features highly affect the fingerprintability across different FP thresholds based on Figure 12a.

Background set. To show the effect of the size of background data on the performance of FP classifiers, we fed 20,000, 40,000, and 80,000 background website instances. According to Figure 11b, feeding more unmonitored traffic instances had little effect on the accuracy and furthermore, the performance of FP classifiers was almost the same against 10-40% FP thresholds.

Since we used class weights to derive both metrics, the imbalance in the number of instances for each label softens. However, considering that the gap between the size of samples in two classes, smallest at the threshold of 70% yielding the worst accuracy, the unbalanced dataset still somewhat affects the classification accuracy. Therefore, to improve the ability of the fingerprintability decision, we suggest closer numbers (e.g.,10,20,90%,etc.) to both ends in the accuracy range for the FP threshold.

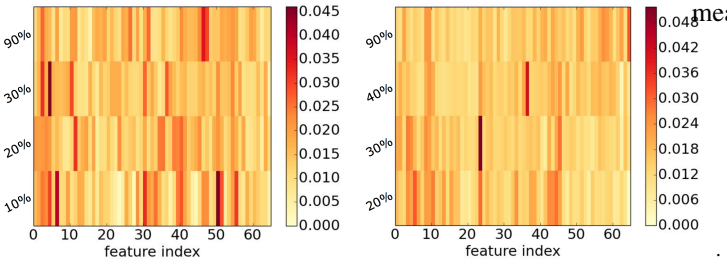
Unpopular websites. To investigate how different testing dataset impacts FP classifiers, we furthermore collected 60 traffic instances of each of 85 unpopular Alexa websites, ranked around 1 million, and downloaded their HTML document files. We trained MLP using *HTML features* of Alexa top 67 websites (60 instances for each) and testing it with HTML features of 5,100 unpopular website traffic instances. In this experiment, we show the performance of FP classifiers toward those unpopular websites.

For good FP thresholds (10-40% and 90%), chosen in previous experiments, we achieved slightly higher error rates



(a) With different confidence thresholds (b) With different sizes of background sets (c) With testing unpopular websites

Fig. 11: Accuracy and MSE when varying the FP threshold and the size of background dataset



(a) Testing with top 50 alexa websites (b) Testing with unpopular 85 alexa websites

Fig. 12: The feature importance of each of 65 features according to FP thresholds.

TABLE XII: Top 15 html properties based feature importance measured by gini index

Rank	Feature
1	total number of unique domains in links
2	total number of third party links
3	total number of unique tags
4	total number of gif files attached in document
5	total number of html files attached in document
6	total number of words in data including script and style
7	proportion of html files attached in document
8	total number of ico files attached in document
9	total number of unique tag paths
10	portion of ico files attached in document
11	total number of domains in links
12	std of number of unique tags per a path
13	proportion of gif files attached in document
14	total number of change of tag path direction
15	total number of characters in data field

while the accuracy was almost same, as shown in Figure 11c. For worse FP thresholds (60-70%), MSE became much greater and the accuracy also decreased. That is, across different sizes of background datasets, the fingerprintability prediction on unpopular websites has more uncertainty with increased error rates and this becomes more severe with bad FP thresholds. However, 10-30% and 90% FP thresholds still yielded trustworthy results with 98-99% of accuracy and 0.1-2% of MSE.

We listed the top 15 features based on feature importance scores for *HTML features* of 85 unpopular websites in Table XIII of Appendix D. As shown in Figure 12b, even though different features appeared in the top list, statistics about domains (feature index 4-5) and embedded web contents (feature index 43) still impact the fingerprintability of unpopular websites, which is also consistent with previous results using Alexa top 100 websites.

Labeling with results of k-FP. In this experiment, we show that our FP classifiers can be applicable to state-of-the-art WF attacks. To generate training labels, we ran *k*-FP [12] classifiers with our dataset consisting of Alexa top 67 monitored websites and 20,000 background dataset. To obtain testing labels, we applied the same experiment setting except that we used 82 unpopular Alexa website for monitored dataset. Then, we extracted 65 *HTML features* based on elements of their HTML documents and based on results, returned by *k*-FP classifiers, we computed *accuracy* for each monitored website and labelled features with 1 or 0 in a way we described earlier. Finally, we trained MLP classifiers with 67 popular website and tested it with 82 unpopular websites. With 30% FP threshold, we achieved 98% of accuracy and 2% MSE. This indicates that our FP classifiers can be further deployed for WF defenses against state-of-the-art WF attacks. We will leave further investigation on other WF attacks for future works.

Deployment. Based on FP classifiers, we can guide developers with safer design of websites against traffic analysis. Furthermore, we can develop a fuzzing like tool to 1) automatically scrutinize documents to identify vulnerable patterns in HTML DOM, which makes website traffic more distinct, 2)propose more resilient HTML DOM, and 3)test new DOM to check if it actually minimizes the predicted fingerprintability. This can prevent the website from leaking the web visiting activity of the user and endangering the onion service

VII. Conclusion

We extensively explored DNN for traffic analysis and proposed 3 different applications, feature engineering, diverse fingerprinting attacks, and fingerprintability predictors. As the feature extractor, the lower representations, learned by an

autoencoder, led competitive results for state-of-the-art WF attacks. As fingerprinting attackers, DNN performed nicely across various different traffic datasets, different fingerprinting tasks, and recent WF defenses. Lastly, MLP classifiers successfully decided if the website is fingerprintable using HTML DOM-based features, suggesting the applicability of DNN models to WF defenses and ensure safer onion service.

REFERENCES

[1] "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[2] "Does alexa have a list of its top-ranked websites ? - alexa support," https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites-, 2015, accessed: 2015-12-14.

[3] "Tensorflow," https://www.tensorflow.org/, 2015, accessed: 2015-12-14.

[4] "Tflearn," http://tflearn.org/, 2015, accessed: 2015-12-14.

[5] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," *ICML Unsupervised and Transfer Learning*, pp. 37–50, 2012.

[6] Y. Bengio and Y. LeCun, "Scaling Learning Algorithms towards AI," 2007. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/bengio-lecun-07.pdf

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

[8] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 227–238. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660362

[9] L. Cun, Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network." [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf

[10] 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 *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, ser. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 332–346. [Online]. Available: http://dx.doi.org/10.1109/SP.2012.28

[11] M. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)âĂŤa review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, aug 1998. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1352231097004470

[12] J. Hayes and G. Danezis, "k-fingerprinting: a Robust Scalable Website Fingerprinting Technique."

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," pp. 770–778, 2016. [Online]. Available: http://www.cv-foundation.org/openaccess/content{_}cvpr{_}2016/html/He{_}Deep{_}Residual{_}Learning{_}CVPR{_}2016{_}paper.html

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, 2006. [Online]. Available: http://science.sciencemag.org/content/313/5786/504

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," feb 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[16] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," dec 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[17] S. Lawrence, C. Giles, Ah Chung Tsoi, and A. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/554195/

[18] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel, "The Variational Fair Auto Encoder," *Iclr*, pp. 1–11, 2016. [Online]. Available: http://arxiv.org/abs/1511.00830

[19] A. S. Miller, B. H. Blott, and T. K. Hames, "Review of neural network applications in medical imaging and signal processing," *Medical & Biological Engineering & Computing*, vol. 30, no. 5, pp. 449–464, sep 1992. [Online]. Available: http://link.springer.com/10.1007/BF02457822

[20] S. E. Oh, S. Li, and N. Hopper, "Fingerprinting Keywords in Search Queries over Tor," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 171–190. [Online]. Available: https://petsymposium.org/2017/papers/issue4/paper59-2017-4-source.pdf

[21] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website Fingerprinting at Internet Scale," *16th NDSS (NDSS 16)*, pp. 143–157, 2016.

[22] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," 2011. [Online]. Available: http://lorre.uni.lu/{~}andriy/

[23] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," jan 2014. [Online]. Available: http://arxiv.org/abs/1401.4082

[24] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated Feature Extraction for Website Fingerprinting through Deep Learning." [Online]. Available: https://arxiv.org/pdf/1708.06376.pdf

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learining Internal Representations by Error Propagation," pp. 318–362, 1986.

[26] R. Schuster, E. Tromer, and V. Shmatikov, "This paper is included in the Proceedings of the 26th USENIX Security Symposium Beauty and the Burst: Remote Identification of Encrypted Video Streams Beauty and the Burst: Remote Identification of Encrypted Video Streams." [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schuster

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," sep 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[28] W. Song and J. Cai, "End-to-End Deep Neural Network for Automatic Speech Recognition." [Online]. Available: https://cs224d.stanford.edu/reports/SongWilliam.pdf

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: https://www.cs.toronto.edu/{~}hinton/absps/JMLRdropout.pdf

[30] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving Neural Network Agents in the NERO Video Game." [Online]. Available: https://pdfs.semanticscholar.org/cdf7/7724e9b7c19b9d3332dd095a4fdc58484022.pdf

[31] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 143–157, 2014.

[32] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor." [Online]. Available: http://www.cypherpunks.ca/{~}iang/pubs/webfingerprint-wpes.pdf

[33] ——, " Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks ," *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[34] Z. Wang, "The Applications of Deep Learning on Traffic Identification." [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Wang-The-Applications-Of-Deep-Learning-On-Traffic-Identification-wp.pdf

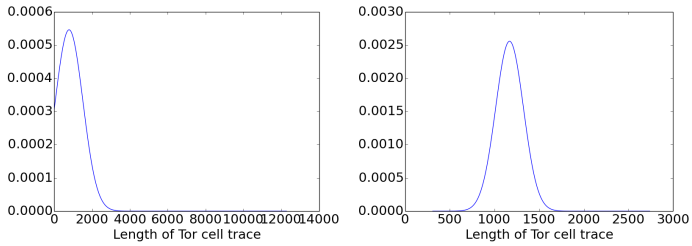
APPENDIX

A. Distribution of sizes of website and keyword traces

We extracted Tor cell sequences for 9,000 website traces and 10,000 keyword traces, and derived their distribution to select candidate dimensions for experiments in Section III-B to determine the optimal dimensions for DNN classifiers.

B. HTML features

- 1. total number of links
- 2. total number of links from same domain
- 3. total number of third party links
- 4. total number of domains in links
- 5. total number of unique domains in links
- 6. total number of tag paths
- 7. total number of unique tag paths
- 8. sum of number of unique tags per a path



(a) 9,000 website traces (median=575, max=10368)

(b) 10k keyword traces (median=1166, max=2738)

Fig. 13: Distribution of size of tor cell traces.

- 9. median of number of unique tags per a path
- 10. mean of number of unique tags per a path
- 11. std of number of unique tags per a path
- 12. total number of change of tag path direction (if depth increases, positive, otherwise, negative)
- 13. total number of non change of tag path direction
- 14. total number of positive direction in tag paths
- 15. total number of negative direction in tag paths
- 16. total sum of tag depths
- 17. std of tag depths
- 18. total number of max depth in tag paths
- 19. total number of min depth in tag paths
- 20. total number of median depth in tag paths
- 21. total number of rounded mean depth in tag paths
- 22. total number of 30% percentile of depth in tag paths
- 23. total number of 70% percentile of depth in tag paths
- 24. max depth of tag paths
- 25. min depth of tag paths
- 26. median depth in tag paths
- 27. rounded mean depth in tag paths
- 28. 30% percentile of depth in tag paths
- 29. 70% percentile of depth in tag paths
- 30. total number of tags
- 31. total number of unique tags
- 32. total number of comments
- 33. total number of attributes
- 34. total number of unique attributes
- 35. total number of characters
- 36. total number of characters in script tag
- 37. total number of characters in style attribute
- 38. total number of characters in attribute
- 39. total number of characters in data including those in script and style attributes
- 40. total number of characters in data attribute
- 41. total number of words in data including those in script and style attributes
- 42. total number of words in data attribute
- 43. total number of image tags in HTML DOM
- 44. portion of image tags over all tag paths
- 45. total number of png files in HTML DOM
- 46.portion of png files in HTML DOM
- 47. total number of ico files in HTML DOM
- 48. portion of ico files in HTML DOM
- 49. total number of jpg files in HTML DOM
- 50. portion of jpg files in HTML DOM
- 51. total number of gif files in HTML DOM
- 52. portion of gif files in HTML DOM
- 53. total number of bmp files in HTML DOM
- 54. portion of bmp files in HTML DOM
- 55. total number of html files in HTML DOM
- 56. portion of html files in HTML DOM
- 57. total number of css files in HTML DOM
- 58. portion of css files in HTML DOM
- 59. total number of js files in HTML DOM
- 60. portion of js files in HTML DOM
- 61. total number of mp3 files in HTML DOM
- 62. total number of avi files in HTML DOM
- 63. total loading time in a capture (pcap) file
- 64. size of a html document file
- 65. size of a capture file

C. Impact of feature dimensions for WF over TLS-encrypted traffic

Based on inter-packet timing sequences, we varied feature dimensions to show how this factor impacts the performance of MLP classifiers and find the optimal dimension. Based on Figure 14, we chose 1,000 and used it for all experiments in Section VI-D.

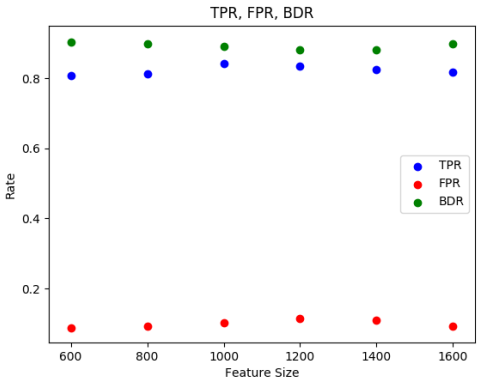


Fig. 14: The performance of MLP classifiers when varying the feature dimension.

D. Top 15 features to predict the fingerprintability of unpopular websites

We computed the sum of feature importance score for each of 65 HTML features and selected top 15 features, which are more important to predict the fingerprintability of Alexa 85 unpopular websites. As mentioned in Section VI-F, features based on domains and embedded web contents still impact the fingerprintability of websites.

TABLE XIII: Top 15 html properties based feature importance measured by gini index when using 85 unpopular websites

Rank	Feature
1	maximum depth of tag paths
2	median of number of unique tags per a tag path
3	mean of number of unique tags per a tag path
4	total number of domains in HTML DOM
5	total number of png files attached in HTML DOM
6	total number of tag paths in HTML DOM
7	proportion of image tags over all tag paths
8	total number of image tags in HTML DOM
9	total number of unique domains in links
10	total number of unique attributes in HTML DOM
11	std of number of unique tags per a tag path
12	std of number of unique tags per a tag path
13	total number of characters in data including script and style attributes
14	total number of links in HTML DOM
15	proportion of gif files attached in HTML DOM