Se Eun Oh*, Nate Mathews, Mohammad Saidur Rahman, Matthew Wright, and Nicholas Hopper

GANDaLF: GAN for Data-Limited Fingerprinting

Abstract: We introduce Generative Adversarial Networks for Data-Limited Fingerprinting (GANDaLF), a new deep-learning-based technique to perform Website Fingerprinting (WF) on Tor traffic. In contrast to most earlier work on deep-learning for WF, GANDaLF is intended to work with few training samples, and achieves this goal through the use of a Generative Adversarial Network to generate a large set of "fake" data that helps to train a deep neural network in distinguishing between classes of actual training data. We evaluate GANDaLF in low-data scenarios including as few as 10 training instances per site, and in multiple settings, including fingerprinting of website index pages and fingerprinting of non-index pages within a site. GANDaLF achieves closed-world accuracy of 87% with just 20 instances per site (and 100 sites) in standard WF settings. In particular, GANDaLF can outperform Var-CNN and Triplet Fingerprinting (TF) across all settings in subpage fingerprinting. For example, GANDaLF outperforms TF by a 29% margin and Var-CNN by 38% for training sets using 20 instances per site.

DOI 10.2478/popets-2021-0029
Received 2020-08-31; revised 2020-12-15; accepted 2020-12-16.

1 Introduction

Tor is one of the most widely used anonymous networks, with eight million daily users [14]. While Tor provides anonymity against basic traffic inspection, it has been shown that more sophisticated attacks can be used to recover some information about Tor traffic. One of the most well-studied attacks against Tor is Website Fingerprinting (WF), which uses the patterns of traffic sent and received on a Tor connection to identify whether the connection is to a list of possible destinations.

WF attacks in the literature have evolved to achieve very high accuracy in classifying known websites. Initial work explored a variety of classification algorithms and useful features to extract from a Tor traffic trace. Through these efforts, WF attacks based on domain-specific classifier models and domain-specific feature extraction [6, 17, 30] reached over 90% accuracy.

More recent work has sought to apply methods from Deep Learning to automate the process of selecting classifiers and extracting features from a network trace [16, 21, 22, 25]. By training classifiers on a much larger data set, using raw traffic traces, and applying techniques to search for the best classifiers within a set of models, this work has led to the discovery of classifiers with over 95% accuracy, and which automatically perform feature extraction without requiring prior domain knowledge.

Attaining superior classification performance with these models, however, is only feasible when using a huge training set, such as 800 samples per site. As Juarez et al. [11] pointed out, most websites are regularly updated and modified, requiring the attacker to frequently collect new training traces for each site. This leads to questions about the feasibility of WF attacks for all but the most powerful attackers.

To demonstrate that WF attacks are a serious threat that even less powerful adversaries can use to undermine the privacy of Tor users, researchers have further investigated more advanced deep learning models that can be optimized in scenarios with limited amounts of training data. Var-CNN [4] adapts the ResNet [7] network model with dilated convolution for traffic analysis with relatively smaller training sets. Triplet Fingerprinting (TF) [26] applies triplet networks with N-shot learning to learn a WF feature extractor that can be used for classification with only a few training samples.

In this paper, we pursue the same objective, *data-limited fingerprinting*, but using generative adversarial networks (GANs) to achieve even better performance for realistic low-data training. Compared to previous

***Corresponding Author: Se Eun Oh:** University of Minnesota, E-mail: seoh@umn.edu
Nate Mathews: Rochester Institute of Technology, E-mail: nate.mathews@mail.rit.edu
Mohammad Saidur Rahman: Rochester Institute of Technology, E-mail: saidur.rahman@mail.rit.edu
Matthew Wright: Rochester Institute of Technology, E-mail: matthew.wright@rit.edu
Nicholas Hopper: University of Minnesota, E-mail: hoppernj@umn.edu

cc BY-NC-ND

work [4, 26], our approach is more efficient, requiring fewer samples than Var-CNN, and offering better performance than TF without the need for pre-training.

To support training with limited data, we propose the novel GANDaLF architecture, which leverages a small amount of labeled data and a larger unlabeled set to train two networks, a *generator* and a *discriminator*. In our setting, the generator is trained to convert random seeds into fake traces from the same statistical distribution as the training data, while the discriminator is trained to correctly classify the labeled data while also discriminating between real traces and fake traces output by the generator. To the best of our knowledge, this represents the first application of *semi-supervised* learning using a GAN in a WF attack. We show that the GANDaLF approach leverages the generator as an additional source of data to help improve the performance of the discriminator, which enables WF to be effective even in low-data settings.

In addition, our work is the first to examine the performance of low-data WF attacks in a more realistic WF scenario, in which users visit both *index* and *non*-index webpages. We present how the generator effectively generates fake traces resulting in a better discriminator (that is, classifier) even when the *intra*-class variance is much larger than in the standard WF scenario. Note that this second scenario makes GANDaLF more promising, especially when the corpus of site subpages is enormous and requires some limited selection of training subpages. We show that the discriminator can be trained even with very few subpages and work even on previously unseen subpages.

We summarize our contributions as follows:

- ***WF with GANs.*** Ours is the first study to investigate the applicability of semi-supervised learning with GANs to WF models. In this paper, we introduce GANDaLF, an extension of the SGAN model of Saliman et al. [23] that is optimized for website fingerprinting. Our modifications include adding deeper generator and discriminator networks, applying regularization techniques in a different way, and adopting improved feature matching loss [13]. In addition, we empirically study the choice of hyperparameters and investigate the optimal labeled and unlabeled data settings that result in a more effective WF attack.
- ***Subpage Fingerprinting Study.*** We study the applicability of GANDaLF and other recent data-limited WF attacks to two WF scenarios, WF with index pages (WF-I) and WF with both index and non-index *subpages* (WF-S). Although other researchers [15, 17] have explored WF-S in the past, to the best of our knowledge, our study is the largest subpage fingerprinting analysis to date and sheds light on the potential of GAN-based fingerprinting. In particular, using the largest website subpage dataset, we revisited state-of-the-art WF classifiers, including traditional machine-learning-based WF, deep-learning-based WF, and more recently published data-limited fingerprinting techniques in both the closed- and open-world scenarios.
- ***Enhanced Low-Data Training.*** In our closed-world lowest data setting using five instances, GANDaLF outperforms Var-CNN [4] by a +40% margin in the WF-I scenario. Even though TF becomes more powerful using five instances in the closed-world WF-I scenario, TF performed poorly in the open-world setting, showing that using the generator of GANDaLF as another data source of "monitored" traces may lead to better generalization in a real-world scenario than the label-based pre-trained network used in TF. Besides the performance improvement, GANDaLF has significant advantages over TF. Unlike the pre-training approach of TF, which still requires regular collection of a large labeled dataset, GANDaLF uses unlabeled data that can often be collected from the same vantage point as used to collect data for the attack itself by running a Tor guard or middle node.

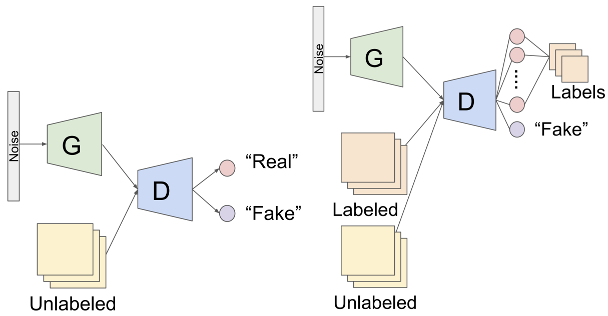
  In the WF-S scenario, GANDaLF outperforms both Var-CNN and TF in all settings by +20-30% and +16-43% margins, respectively. However, we found that k-FP becomes more effective using more limited training data (i.e., 5-20 instances) and in the open-world setting, since feature engineering based on packet statistics is better able to handle large intra-class variance in our subpage dataset. We show that manual feature engineering is helpful in WF-S, while GANDaLF outperforms all other DL-based classifiers.

2 Background

In this section, we first organize WF attacks into three categories: WF models based on traditional machine learning techniques, techniques based on Deep Neural Networks (DNNs), and finally advanced techniques that specialize in maximizing performance in low-data scenarios. Finally, we discuss GANs.

2.1 Website Fingerprinting

Website fingerprinting is a traffic analysis technique in which the adversary aims to identify the websites that



(a) Vanilla GANs **(b)** Semi-supervised learning with GANs

Fig. 1. Generative adversarial networks (GANs).

a user is visiting. To perform this attack, the adversary first makes a series of connections to sites of interest and saves the traffic patterns created to create a labeled dataset consisting of pairs: (`traffic_pattern`, `website`). He then uses this dataset to train a machine learning classifier to recognize these websites based on traffic patterns observed from the victim's online activity. This attack can be successful because the observable traffic patterns across multiple visits to the same webpage are relatively consistent and often fairly distinct from the patterns seen when visiting other sites. Because these attacks depend solely on traffic metadata (rather than the traffic contents), WF attacks remain successful even when traffic is encrypted or under the protection of privacy enhancing technologies such as VPNs or Tor.

Closed- and Open-World Settings. An important consideration in the study of WF is whether an evaluation uses a *closed-world* setting or an *open-world setting*. In a closed-world setting, which is typically used to make baseline comparisons between models, the victim is assumed to be visiting one of a fixed set of sites that the attacker is interested in and can train on, known as the *monitored set*. In contrast, an experiment in the open-world setting also uses a large set of sites outside of the monitored set, known as the *unmonitored set*, and allows the victim to visit a site in either set. Since in practice, a user could be visiting any site on the web, open-world evaluations model is a more realistic scenario. It should be noted, however, that open-world evaluations cannot test every possible unmonitored site on the Web – attacks tend to degrade in accuracy as larger unmonitored sets are used in an evaluation. Additionally, neither scenario considers the rate at which users actually visit different sites while using Tor (the *base rate* of each site). Attacks will be more accurate in practice when including popular sites in the monitored set than when looking for less popular sites.

2.1.1 Traditional ML Attacks on Tor

The first WF attack on Tor was introduced by Herrmann et al. [8] using a Naive Bayes Classifier based on the packet length frequency. After that, Panchenko et al. [18] further improved WF performance by using a Support Vector Machine (SVM) and investigating various feature sets. Since then, other researchers [6, 17, 30] explored different classifiers and new feature sets to develop more effective WF models:

***k*-NN.** Wang et al. [30] adapted a *k*-Nearest Neighbor (*k*-NN) classifier for effective WF with a large feature set, and attained 95% accuracy in a closed-world setting with the index pages of 100 websites.

CUMUL. Panchenko et al. [17] significantly improved the performance of the SVM classifier with a new feature set, called CUMUL, that is based on cumulative sizes of TCP packets, lengths of TLS records, and numbers of Tor cells.

***k*-Fingerprinting.** Hayes and Danezis [6] proposed the *k*-fingerprinting (*k*-FP) attack, which leverages a Random-Forest classifier and a statistics-based feature set to achieve high performance. In their open-world evaluation, they computed the hamming distances between RF leaves and used these in a *k*-NN classifier. This allowed them to tune their attack towards either high precision or high recall by varying the value of *k*.

2.1.2 Deep-Learning-Based Attacks

WF researchers [16, 22, 25] began investigating the applicability of deep learning (DL) to automate the feature extraction process in WF:

Automated Website Fingerprinting (AWF). Rimmer et al. [22] examined various DNN models such as stacked-denoising autoencoders, convolutional neural networks (CNN), and Long Short-Term Memory (LSTM). For the evaluation, they collected the largest dataset of traffic traces in the WF literature. Their best attack reaches up to 94% accuracy when evaluated in a closed world of 900 websites. It requires 2,500 training samples per website to achieve this, however. In our paper, we used the dataset they collected to evaluate GANDaLF.

Deep Fingerprinting. Sirinam et al. [25] proposed Deep Fingerprinting (DF), a CNN model adapting recent advances in computer vision to create a deeper and more effective network. The DF model got 98% accuracy in a closed world of 95 websites, over 90% accuracy for traffic traces defended by WTF-PAD [12] – a defense considered the lead candidate for deployment in Tor [19], and 98% top-two accuracy on traffic defended by Walkie-Talkie [31]. To get this level of accuracy, however, they require 800 training samples per website.

2.1.3 Low-Data WF

The AWF and DF models were trained using hundreds of samples for every website in the monitored set. Sirinam et al. noted that collecting such large datasets would require an attacker to run between 8-24 PCs continuously [26], where the dataset needs to be refreshed every few weeks at least to maintain high accuracy [11, 22]. Given this, one can criticize the attacks as being unrealistic except for powerful adversaries who could use the resources to instead perform other attacks on Tor. To address this, researchers have begun focusing on the design of WF attacks that overcome the cost and time to obtain training data. Two recent works [4, 26] focus on the optimization of DNN architectures to achieve state-of-the-art performance when few fresh samples are available for training. These attacks allow hypothetical adversaries to quickly apply ready-to-use models, even when subjected to such constraints.

Var-CNN. To achieve comparable classification performance in low-data scenarios, Bhat et al. [4] introduced Var-CNN, an optimized DNN architecture based on ResNet [7] with dilated causal convolutions. They also proposed to combine direction and timing information together. Those improvements enabled their model to get 97.8% accuracy using just 100 training instances per website across 100 websites.

Triplet Fingerprinting. To achieve high performance with as few fresh training samples as possible, Sirinam et al. [26] pursued the concept of *N-shot learning* with their Triplet Fingerprinting (TF) attack. For this attack, the adversary first pre-trains a feature extractor using large, publicly available training sets such as the AWF dataset [22]. The feature extractor is trained to produce a vector as output that captures the feature information in a way that minimizes the distances between traces from the same website and maximizes the distances between traces from different websites. Using the features derived from this feature extractor, a small dataset of fresh samples are then used to train and test a simple distance-based classifier, such as k-NN. This attack was shown to reach 94% accuracy when only using 10 instances per website to train the classifier.

Although Sirinam et al. addressed the problem of data-limited fingerprinting, it requires pre-training a model using a large labeled training dataset. While such datasets are available, such as the AWF dataset collected in 2016, Sirinam et al. found a significant loss in performance when using data from three years before [26]. It is likely that the attacker would still need to regularly update the labeled dataset to maintain a high performance level. In contrast, GANDaLF uses a large unlabeled dataset. This dataset can be obtained by eavesdropping on potential victims *from the same vantage points as used to perform the attack*. For example, the attacker could run a Tor guard or middle node or use malware to compromise the DSL modems through which users connect to the Internet. From these vantage points, they can collect unlabeled data to train the GAN as well as the data for performing the actual WF attack, without significant additional cost. As such, to properly update WF models using a fresh dataset, the use of unlabeled data can make WF attacks more portable while keeping high performance.

2.1.4 WF with Subpages

For the majority of the experiments performed in the aforementioned works, researchers have used a website's *index* page to represent the site (e.g. going to http://www.cnn.com/ by typing it into the browser's URL bar). This attack strategy limits the applicability of the attack as it severely limits the amount of traffic instances that may be viably fingerprinted in the real-world. Panchenko et al. were the first to identify this issue [17]. An attacker would be interested to instead identify a website using traffic samples generated from *any* webpage on the site (e.g. news stories on CNN.com reached by following links from the homepage or social media). In this scenario, the attacker can use both index pages and *non-index* pages (also referred to as *subpages*) for website identification. To distinguish between these settings, we use the phrases *WF with index pages* (*WF-I*) and *WF with subpages* (*WF-S*).

When Panchenko et. al first studied this scenario, they used a small dataset of 20 websites, each represented by 51 subpages, for which up to 15 samples were collected (i.e. 20x51x15). In this paper, we extend this study further by collecting a larger dataset of size 24x90x90 and perform a thorough comparison of WF attacks in this setting.

Tangentially, Oh et al. studied the fingerprintability of keyword search queries for popular search engines when visited over the Tor network [15]. This work is similar to that of WF-S, since the webpage generated by each keyword search query represents a unique subpage of the search engine website domain. Our work is different, however, in that each search term's subpages are identified as their own class, allowing for the identification of individual pages within a domain. For non-search cases, this would be equivalent to trying to classify the subpages separately (e.g. https://www.cnn.com/politics as different from https://www.cnn.com/business) instead of as members of the same class (e.g. for CNN.com in general). We use the term *subpage identification* to describe this type of attack scenario. For this paper, we have limited our study to WF-S and do not explore subpage identification.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced in 2014 by Goodfellow et al. for creating images [5]. Figure 1a depicts a vanilla GAN, which consists of two components: a *discriminator* and a *generator*. The discriminator is a two-class classifier that is trained to distinguish between real data samples and fake data samples that are generated by the generator. The generator is trained to create those fake data samples in a way that is as hard to distinguish from the real samples as possible. If both models are successfully trained, the generator will become good at producing samples that are indeed very similar to the original data distribution, such as realistic-looking images.

The discriminator uses a mostly standard convolutional neural network (CNN) that takes the image as input and outputs a single value in the range $[0, 1)$ that can be interpreted as a measure of how real the input image is, i.e. the likelihood that it comes from the distribution of real data, $p_{data}(x)$. The generator, on the other hand, is given a noise vector $p_z(z)$ as input and effectively operates as a reverse CNN, repeatedly using a method called *deconvolution* to transform the noise into image features and create an output in the shape of an image.

The vanilla GAN thus alternately minimizes two different loss functions to achieve these two different goals:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1\text{-}D(G(z)))]$$

In words, the discriminator, D, is trained to maximize the probability of assigning the correct label to both original and reconstructed samples from the generator, G, while G is trained to minimize $\log(1 - D(G(z)))$.

Table 1. The data setup for GANDaLF in Section 6 (Note that ( ): the number of instances per class, $\mathbf{L}$: labeled set, $\mathbf{U}$: unlabeled set, $\mathbf{GF}$: GDLF, $\mathbf{n}$: $\{5, 10, 20, 50, 90\}$).

	WF-I		WF-S	
	CW	OW	CW	OW
L	AWF1 (n)	AWF1&AWF-OW (n&100×n)	GF25 (n)	GF25&GF-OW (n&25×n)
U	AWF2 (2500)	AWF2&AWF-OW (2500&33k)	AWF1 (2490)	AWF1&AWF-OW (2490&400k)

3 Data

In this section, we discuss the details of the datasets used for our attack evaluation in Section 6. The datasets we use are organized into two different types: index-page only data for use in WF-I, and datasets containing a mix of subpages to be used for WF-S.

3.1 Index Webpage Set

For the WF-I scenario, we use the large datasets collected by Rimmer et al. in 2016 [22]. To the best of our knowledge, they provided the largest dataset, comprising 2,500 instances of each of 900 monitored websites and 400,000 unmonitored websites. The sites were chosen among top Alexa websites [2].

To produce the dataset used for our evaluations, we chose 200 websites randomly sampled from their monitored dataset. We further partitioned this data into two distinct sets containing samples for 100 websites each, which we denote **AWF1** and **AWF2**. For our WF-I experiments, we use AWF1 as our monitored dataset and use the AWF unmonitored dataset (**AWF-OW**) as is. We then use the AWF2 dataset to act as the unlabeled dataset when training GANDaLF. This is also the dataset we use to pretrain the TF attack. Furthermore, to investigate the impact of labeled data and unlabeled data on the performance of GANDaLF , we adopt DF set (**DF**) provided by Sirinam et al. [25] that was collected using 40 circuits and labelled with the circuit index along with the website class.

All traces in AWF and DF set consisted of the packet direction information in which each column was marked as -1 if it is an incoming packet and 1 otherwise.

3.2 Subpage Set

In order to evaluate the effectiveness of subpage fingerprinting we needed to collect a new dataset using `tor-browser-crawler` [3]. Before beginning the data crawl, we first harvested a list of non-index URLs for websites sampled from the Alexa top 200 rankings.

We then downloaded those websites locally using `torsocks wget` and identified candidate URLs using the `find` command. Finally, we eliminated domains that had low subpage counts and trimmed our final list of URLs to subpages within 25 domains.

Next, we randomly selected an equal number of non-index pages for each website, which we visited many times in a round-robin fashion. After collecting the traffic for each visit, we filtered out traces that failed to load and that contained less than 150 packets. We then trimmed the dataset such that the number of non-index pages and samples per non-index page were equal for every examined website. This process left us with 39 instances per subpage and 96 subpages per website, for a total of 3,744 instances for each of our 25 websites. We will refer to this dataset as **GDLF25** from here on out.

For WF-S CW experiments, we use all 25 domains in GDLF25 as our monitored dataset. To evaluate WF-S in the OW setting, we further crawled the unmonitored subpage dataset using urls provided by other researchers [22], leading to 70,000 subpages (**GDLF-OW**). Finally, we use the AWF1 dataset to act as the unlabeled and pretraining data for the GANDaLF and TF attacks. Table 1 summarizes the dataset usage for two different scenarios: WF-I and WF-S.

4 Semi-Supervised Learning with GANs

Given that GANs have achieved success for various *generative* applications, researchers have become interested to see if they can also help in *classification* tasks as well. In particular, GANs have been applied to the problem of *semi-supervised learning (SSL)* [27, 28]. The goal of SSL with GANs is to transform the discriminator into a multi-class classifier that learns from a relatively small amount of labeled data (the supervised part) while also learning from a larger body of unlabeled data (the unsupervised part). As shown in Figure 1b, this classifier (the discriminator D) outputs not only a single value that measures whether the input was real (closer to 1) or fake, but also it outputs a set of K individual class probabilities that can be used to identify the label for any real samples. The generator G is trained as before only to produce samples that appear real without necessarily fitting any of the classes in particular, which eventually helps classify the dataset.

The benefit of this approach is that it can achieve high performance on the classification task while only using a relatively small sample of labeled data. In many tasks, unlabeled data is readily available, but obtaining large quantities of data with accurate labels is often expensive. The insight of using a GAN is that, during the process of learning how to discriminate real samples from fake ones, the discriminator is also learning how to extract meaningful features from the data by utilizing a large amount of unlabeled data. Having built up this feature extraction capability, the model is then able to learn how to distinguish between different sites with relatively few samples. Indeed, Saliman et al. [23] showed the potential of this idea by improving the performance of a classification task using GANs in this way. In this paper, the overall GANDaLF design is based on their GAN, which is called SGAN. We next provide a brief overview of the SGAN design.

4.1 SGAN Overview

SGAN is composed of a generator and a discriminator, which are trained together in the same way as a vanilla GAN. The key difference from the vanilla GAN is in the loss functions of the generator and discriminator. Saliman et al. [23] introduced a new loss function, called *feature matching loss*, to be used as the *generator loss*. We will discuss this loss function in detail in Section 4.2. The *discriminator loss* is a combination of the *supervised loss*, based on how well it classifies labeled inputs into the K classes, and *unsupervised loss*, based on how well it can distinguish real inputs from the unlabeled dataset and the fake inputs from the generator.

The original concept of the supervised loss was to optimize the classification over $K + 1$ labels where K is the number of classes in the labeled set, and there is one additional label for fake data. However, since the classifier with K + 1 softmax outputs is overparameterized, as an efficient implementation [1], Saliman et al. suggested to fix the unnormalized logit as 0 for fake data, which in turn, leads the supervised loss to a categorical cross-entropy loss over the softmax output of K classes and the discriminator D to be $D(x) = \dfrac{Z(x)}{Z(x) + \mathsf{exp}(l_{fake}(x))}$ $= \dfrac{Z(x)}{Z(x)+1}$, where $Z(x) = \sum_{k=1}^{K} \mathsf{exp}(l_k(x))$ where l is the output logits. We will detail this unsupervised loss implementation of the discriminator in Appendix A.

4.2 Feature Matching Loss

State-of-the-art GANs have been shown to generate excellent samples for many tasks [20], but training GANs is difficult and very sensitive to the hyperparameter settings. A particular issue is that the two models can end up focusing too much on mistakes the other model is making without actually improving on the core task. For example, if the generator is adding a blue blob to the corner of nearly every image, then the discriminator will heavily emphasize this blob in its decisions, improving its score without improving its function. The generator in turn may learn to change the color of the blob to red without removing it. This can create a fruitless back-and-forth that does not yield good images.

To solve this problem, Saliman et al. [23] proposed a new loss function for the generator, called feature matching loss, to prevent it from overtraining on the current discriminator. Feature matching loss is computed on the output of an intermediate layer of the discriminator, which can be thought of as a representation of the features that the convolutional layers have found. By seeking to minimize the difference between the features found in real data and the features found in its generated samples, the generator can avoid focusing unduly on a single mistake it is making that would be heavily emphasized in the final discriminator output. This strategy makes GANs more stable in training, and we leverage it in GANDaLF as well. In GANDaLF, the feature matching loss was computed for both WF-I and WF-S scenarios by capturing the feature vectors in the flatten layer prior to the last convolutional 1D layer, as shown in Figure 2.

5 GANDaLF

In this section, we introduce our new attack, GANDaLF, starting with the threat model, then a discussion of the intuition behind the choice of SGAN for WF, and present details of its design and network architecture. Furthermore, we emphasize the contribution of GANDaLF based on our experiences and findings when tuning the GAN for WF in the semi-supervised setting and in comparison to prior WF attacks.

5.1 Threat Model

We assume a network-level, passive adversary who can only observe the network traces between the client and the middle node of a Tor circuit, possibly by operating the entry guard or middle node. The attacker is not able to drop or modify packets that have been sent and received from servers or collude with web servers.

The attacker requires training data, but since collecting data from a client-based web crawl can be expensive, this attacker seeks to run an entry guard or middle node to both perform the attack and simultaneously gather live Tor traffic that can be used as unlabeled data. Note that while the middle node position has less direct information about the client, Jansen et al. [10] showed how an attacker can use the middle node to perform attacks such as WF.

The attacker is interested in two attack scenarios, both of which we explore in this paper. The first goal is to train GANDaLF using website index pages and then fingerprint visits to index pages only. This approach has been used by most previous WF research [6, 16, 22, 25]. We call this scenario WF-I, referring to fingerprinting websites with index pages.

The attacker's second goal, which might include more realistic scenarios, is to train GANDaLF using both index and *subpages* (i.e. non-index pages) from a website and then try to identify visits to any subpage of a website. For example, the attacker may want to classify any page of amazon.com as Amazon. Note that the attacker only needs a subset of the subpages from each website instead of all pages to train the model, and can test it using unseen subpages by leveraging the generative ability of GANDaLF. We call this scenario WF-S, referring to fingerprinting websites with index and non-index pages.

We explore WF-I and WF-S scenarios in both closed-world (CW) and open-world (OW) settings. In CW experiments, the attacker keeps a webpage fingerprint database and assumes that users will only visit webpages in this database. In the more realistic OW setting, the attacker keeps a set of *monitored sites* and attempts to classify whether a particular trace is to a site in this set or outside the monitored set. To achieve this, the attacker collects traces of both monitored and unmonitored websites to train the classifier and predicts unseen webpages using this trained model to answer whether or not they are monitored.

5.2 Sources of Unlabeled Data

Several groups of WF researchers [4, 16, 22, 25] have demonstrated the effectiveness of convolutional neural networks (CNNs) to model the distribution of website traces, resulting in WF attacks with high classification accuracy. Based on CNNs, we built an SGAN model with a generator and a discriminator, in which the discriminator becomes a $K+1$ class WF classifier (K is the number of websites in the labeled set). This classifier utilizes three different sources of training data: labeled website traces collected by the attacker, unlabeled websites traces that could be from a publicly available WF database or fresh Tor traffic collected by running entry

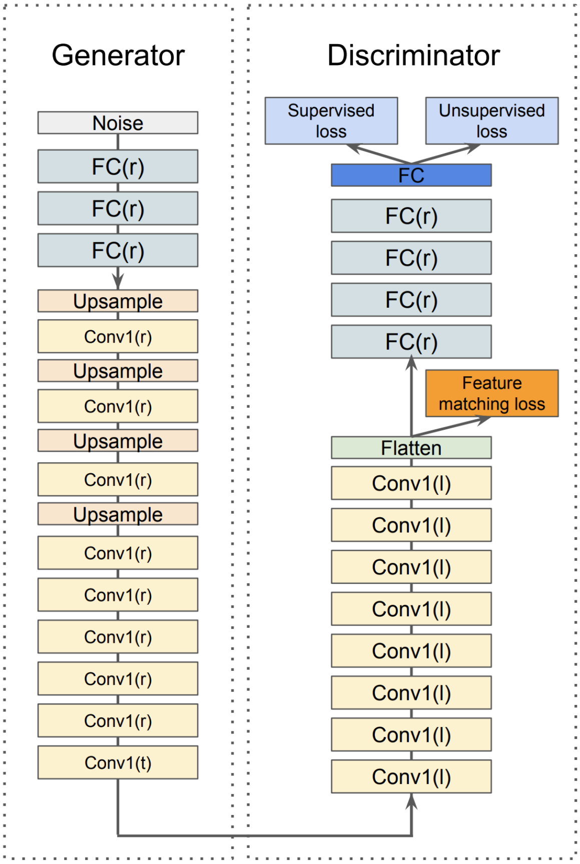


Fig. 2. GANDaLF architecture (**FC**: Fully-connected layer, **Conv**: convolutional layer, **r**: ReLU, **t**: Tanh, and **l**: LeakyReLU). Note that in WF-I, we used one fully connected layer for the generator.

guards or middle nodes, and fake website traces produced by the generator.

The combination of different training sources enables GANDaLF to learn from a broader perspective, which leads to more precise WF classification only using a few labeled samples for training. In contrast, the learning capacity of supervised WF techniques is limited to the data distribution when using a small set of training samples, which leads to significantly weaker performance in the limited-data setting.

Since GANDaLF needs multiple data sources, the choice of unlabeled data impacts its classification performance. SGAN [23] constructed both labeled and unlabeled datasets from the same data distribution. In a WF attack, however, this would require the attacker to collect a very large unlabeled set to be aligned with the labeled data, which contradicts the goal of low-data training. Thus, to investigate the applicability of SGAN to low-data WF, we studied how different the unlabeled data distribution is from the labeled data distribution if we construct them from different datasets.

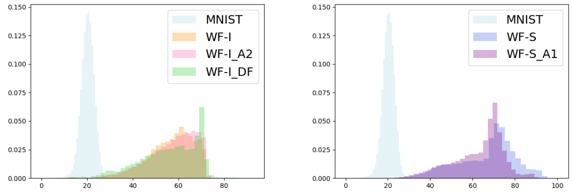


Fig. 3. Distribution of euclidean distances between labeled and unlabeled data (**A1**: AWF [22] set consisting of 100 websites, **A2**: AWF set consisting of 100 websites (different from A2), and **D**: DF set [25]).

We explored three datasets – WF-I, and WF-S, and the MNIST computer vision dataset to serve as a baseline. For MNIST, we built both labeled and unlabeled data from the MNIST set. For WF-I, we constructed the labeled set using the AWF1 set [22] and three different unlabeled sets: (i) one based on the same AWF1 set (WF-I in Figure 3), (ii) one based on the AWF2 set (WF-I-A2), and (iii) one based on the DF set collected with different network settings (WF-I-D). For WF-S, we used GDLF21 to be used as labeled set and build two unlabeled sets: (i) one from the same GDLF21 set (WF-S in Figure 3) and (ii) one from AWF1 (WF-S-A1). Then we computed the pair-wise Euclidean distances between labeled and unlabeled data in these settings; the distributions of these distances are shown in Figure 3.

Figure 3 shows that WF-I-A2 is close to WF-I, which indicates that if the traces are collected in the same network environment, their distance distributions are almost the same, even though they comprise different website traces. In WF-I-D, as the DF set was collected in different network settings, the distances had more variance. Similarly, the distance distributions of WF-S-A1 and WF-S became more different from each other, which we assumed as the most difficult data setup for GANDaLF. We will empirically show in Section 6 the impact of using unlabeled data chosen from other distributions, and conclude that it has minimal impact on the classification accuracy, but it does affect the stability of training and somewhat restricts the capacity of supervised learning.

With this preliminary analysis, we see that SGAN is promising to explore WF in the low-data setting by using a small labeled set together with a large unlabeled set to help train the discriminator. Furthermore, we will study more optimal labeled and unlabeled data settings to maximize the classification power of GANDaLF in Section 6. Compared to MNIST with a normally distributed curve, however, Figure 3 shows more variance in the distances between labeled and unlabeled data for WF data. This means that we need careful tuning of SGAN to ensure better performance, which we discuss in Section 5.3.

In addition, we expect that WF-S is a more challenging task than WF-I, because traces in WF-S are more different than WF-I as they are plotted on a wider curve in Figure 3. This results in additional difficulty to simulate realistic fake subpage fingerprints as well as to classify fingerprints to correct website labels.

5.3 SGAN Optimization for GANDaLF

Saliman et al. [23] proposed several SGAN architectures optimized for different datasets, including MNIST, CIFAR-10, and SVHN. Among these architectures, we selected the one optimized for CIFAR-10 as our starting point, since it yielded better initial accuracy on the AWF1 set.

In this section, we discuss the technical challenges we addressed to find the optimal SGAN architecture for WF tasks and key design decisions. First, we found several aspects of SGAN to be problematic when applied directly to the WF problem.

- SGAN was built based on two-dimensional (2D) convolutional layers. As pointed out by Sirinam et al., however, network traffic features do not carry a meaningful 2D spatial pattern in the same way as the images that most CNNs operate on [25]. Thus, we had to incorporate one-dimensional (1D) convolutional layers into SGAN, and further, tune the model towards WF classification tasks. The application of 1D convolutional layers to SGAN revealed several additional problems that we needed to address to improve the performance of GANDaLF.
- SGAN used neither a batch normalization (BN) nor dropout in the generator. However, building the initial SGAN architecture to use 1D convolutional layers made training unstable. Thus, we explored whether adding BN or dropout layers to both generator and discriminator would help improve the training process.
- Saliman et al. proposed feature matching loss using the mean absolute difference (i.e., L1 loss) between the expected features of real data and the expected features of the generated data. However, since webpage traces are different from image features, we also investigated different feature matching loss functions (L2 vs L1 distance).
- The choice of hyperparameters impacts the performance of SGAN. Thus, we had to empirically find the optimal parameters for WF-I and WF-S, respectively.

To overcome these limitations of the original SGAN architecture, we introduced the following technical innovations. Note that we used the same architecture for WF-I and WF-S, but we empirically selected hyperparameters for each scenario as shown in Table 2.

Deeper 1D-Based Design. The initial SGAN implementation [1] with feature matching was based on the generator containing four deconvolutional 2D layers [1] and a discriminator consisting of seven 2D convolutional layers. After simply switching from 2D convolutional layers to 1D layers, we trained it using 90 instances per website and it reached 78% CW accuracy in the WF-I setting. As shown in Figure 2, we added more 1D convolutional layers, which resulted in a higher accuracy. This change led to a classifier that obtained 95% accuracy with 90 training instances for each of the 100 websites.

Dropout and BN. We found that *selective* use of dropout layers and the full use of BN layers in the generator helps to make the training more stable in WF-S. More specifically, we added a dropout layer after all convolutional layers except the first and last layers as shown in Figure 2. In contrast, for WF-I, we only used BN layers in the generator, since the use of dropout layers in any location worsened the performance. Furthermore, we added several fully connected layers, followed by dropout layers between the flattened layers, where we captured features to compute the feature matching loss, and the last output layer.

Different Generator Loss. We noticed that the same L1 feature matching loss works properly for WF-I, while L2 loss improved the testing accuracy in the WF-S scenario more than L1 loss. However, in both scenarios, generator loss started with very low value around 0 and did not decrease much, while discriminator loss continually decreased. This indicates that the generator did not generate actual good fake traces, while the supervised performance was constantly improved. This is because intra-class variation in WF traces is more significant than for images such as MNIST, which made it harder for GANDaLF to reduce the feature matching loss. Furthermore, when using AWF1 set as unlabeled data in

[1] A deconvolution is the inverse operation of the convolution, which means performing the convolution in the back propagation.

Table 2. Hyperparameter optimization showing the chosen parameters and search spaces for the WF-I and WF-S scenarios (**G**: generator, **D**: discriminator, [**Conv**]: 1D convolutional layer block, [**Full**]: fully-connected layer block, **Up**: Upsampling layer, **act**: activation function, and #: number).

Scenario →	WF-I			WF-S		
HyperParam ↓	Choice		Search Space	Choice		Search Space
	G	D		G	D	
[Conv] layer#	9	8	4∼12	9	8	4∼12
[Conv] filter#	64∼ 512	32∼ 256	10∼1,000	32∼ 256	32∼ 256	10∼1,000
[Conv] filter size	5	5	2∼10	20	20	2∼30
[Conv] stride size	1	1∼2	1∼4	1	1∼4	1∼4
[Conv] dropout rate	-	0.3	0.2∼0.9	0.3	0.3	0.2∼0.9
[Conv] act	ReLU	LeakyReLU	ReLU, LeakyReLU, ELU	ReLU	LeakyReLU	ReLU, LeakyReLU, ELU
[Conv] Up#	4	-	2∼9	4	-	2∼10
[Full] layer#	1	5	1∼6	3	5	1∼5
[Full] node#	316	512∼ 2,048	128∼2,048	316	512∼ 2,048	10∼2,048
[Full] dropout rate	-	0.5	0.2∼0.9	0.5	0.5	0.2∼0.9
[Full] act	ReLU	ReLU	ReLU,LeakyReLU	ReLU	ReLU	ReLU, LeakyReLU
input dim	5,000		5,000	5,000		3,000∼8,000
z dim	100		50∼700	100		50∼700
optimizer	Adam		Adam	Adam		Adam
learning rate	$2e^{-4}$	$5e^{-5}$	$1e^{-5} \sim 0.1$	$2e^{-4}$	$5e^{-5}$	$1e^{-5} \sim 0.1$
epoch	≤30(CW), ≤150(OW)		10∼1,000	≤10(CW), ≤100(OW)		10∼1,000
batch	32		16∼128	16		16∼128

WF-S, the generator loss kept increasing even as the discriminator loss was decreasing. We will investigate this problem in detail in Section 6.3.

Stride and Kernel Choice in WF-S. Furthermore, we found that a greater length of strides and kernels helped improve the performance of GANDaLF in WF-S. This was consistent with our expectation that increasing the stride length and kernel sizes, which shrinks the output volume after the convolutions, might lead the network to better handle WF-S having greater intra-class variation than WF-I and capture meaningful features. This resulted in the number of features used to compute the feature matching loss in WF-I being 20,224, while it was 1,280 in WF-S scenario. As such, losing some details by increasing the stride and kernel sizes helps to better capture the traffic pattern when features are more variable within each class.

Input Representation for WF-S. Most DL-based WF attacks represent a website trace as a sequence of ±1's that indicate packet direction. In our investigations, we explored several alternative data representations, such as inter-packet delay (IPD) and Tik-Tok [21] sequences, for both WF-I and WF-S scenarios. In the WF-S scenario, we found that IPD yielded +9% and +8% better CW accuracy than the direction and Tik-Tok features. Hence, we used IPD sequences in WF-S scenarios throughout the paper.

Parameter Tuning. Along with the architectural tuning, we also explored different combinations of parameters involved in the architecture for WF-I and WF-S. Since the GDLF dataset is different from the AWF dataset, we conducted hyperparameter tuning separately for each scenario. We used 90 instances of AWF1 and all of AWF2 for tuning WF-I, and 90 instances of GDLF25 and all of AWF1 for tuning WF-S. In this way, we can ensure that the overlap is minimal between the tuning sets and the testing sets used in Section 6.

The parameter search space and chosen parameters are reported in Table 2. Beyond these parameters, we also adjusted other components in SGAN. First, the SGAN of Saliman et al. [1] used weight normalization (WN) [24] in the discriminator, while we applied batch normalization since WN barely impacted the performance, and BN is easier to implement. Second, we applied different learning rates to the discriminator and generator during the optimization based on findings by Heusel et al. [9] that this ensures better convergence to Nash equilibrium, and further, led GANs such as DCGAN [20] to achieve better performance.

Summary. Overall, the most effective design for GANDaLF is to go much deeper by adding fully-connected layers and more convolutional layers. As shown by Sirinam et al. [25], more layers help the model learn the inner structure of website traces more effectively since WF set has more inter- and intra-class variances than the image set. On the downside, this may make the model more complicated, resulting in more chances of overfitting. Thus, we added dropout and batch normalization layers to relieve this concern.

6 Evaluation

In this section, we evaluate the performance of GANDaLF in various experimental scenarios. First, we compare GANDaLF to the state-of-the-art WF techniques in the WF-I setting (index pages) with limited training data. Then we further investigate the applicability of GANDaLF and other data-limited attacks in the WF-S setting (subpages).

6.1 Experimental Setting

Setup. We implemented GANDaLF using Tensorflow; each experiment was conducted on a Tesla P100 GPU with 16GB of memory. Using pseudocode, we provide details of the GANDaLF experimental setup in Algorithm 1 of Appendix B. We evaluated each technique using five trials and added more experiments up to a maximum of 20 when the standard deviation was greater than 1%.

To implement state-of-the-art WF techniques, we adopt the original implementations provided by researchers [4, 6, 17, 25, 26]. We made few changes when necessary for the data loading pipeline and for hyperparameter tuning. When tuning k-FP, we explored different numbers of trees from 500 to 2,000 and finally chose 2,000 for both scenarios. For DL-based WF attacks, we explored different mini-batch and convolutional stride sizes, as these are parameters that are significant for GANDaLF. Both DF and TF were also allowed to train for additional epochs until validation loss increased for five consecutive epochs. Since TF [26] used 1-20 training instances per website for the N-shot learning, we chose similar training set sizes, however, we increased the size up to 90 instances to see how much DL-based classifiers are benefited by additional training instances. That is, to construct the training labeled set, we randomly sampled 5, 10, 20, 50, and 90 instances for 100 websites in WF-I. In WF-S (subpages), we randomly chose one instance using each of n_s subpages per site in which n_s = 5, 10, 20, 50, and 90 (i.e., total $1 \times n_s$ instances).

For GANDaLF, we randomly sampled these instances rather than using one subpage per site, since this approach yielded slightly higher closed world accuracy, which will be detailed in Section 6.3. For other classifiers, we chose $1 \times n_s$ achieving a higher accuracy. In either case, the standard deviations between trials in WF-S are greater than WF-I, most likely due to much larger intra-class variance.

Table 3. WF-I, CW: Comparison to k-FP, DF, Var-CNN, and TF using 5-90 training instances. We do not show standard deviations less than 1%. We measured the time (s: seconds) for testing 42k testing samples. Other numbers are %.

TrainN	GANDaLF	k-FP	DF	Var-CNN	TF
5	70±2	61	60±2	25.9	78±1
10	81±1	72.5	79±2	69.1	81.6
20	87±1	77.3	89±2	90.8	83.1
50	93±1	82.8	95.1	97.1	83.9
90	95±1	85.5	97.1	98.3	84.2
time	5.5s	1.1s	7.6s	43.6s	8.5s

Metrics. We summarize the metrics for CW and OW evaluation as follows.

– *Accuracy*: The percentage of predictions that are correct. This metric is traditionally used to evaluate classifiers in the CW setting in prior WF work, which we adhere to.
– *Precision*: The percentage of positive predictions (i.e. predicted as "monitored") that are correct. If the classifier is tuned for high precision, it minimizes the number of users being misdetected as "guilty," but may miss some instances that were truly monitored.
– *Recall*: The percentage of monitored-site instances that are classified as "monitored." A classifier tuned for high recall will reliably identify when a sensitive site has been visited, but may also misidentify "innocent" websites as sensitive.

A WF adversary must consider both the precision and recall of their classifier when evaluating the results of a real-world attack, so we show precision-recall curves for our OW experiments.

6.2 Fingerprinting Websites with Index Pages

In this section, we evaluate the classification ability of GANDaLF and other WF techniques in a low-data setting by training and testing with website index pages.

CW Performance. We trained GANDaLF, k-FP, DF, Var-CNN, and TF classifiers using 5-90 instances per website, randomly sampled from the AWF1 dataset. To train GANDaLF, we used AWF2 as the unlabeled data. For a fair comparison, we also used the AWF2 dataset for the pre-training phase of the TF attack. The

Table 4. WF-I, CW: Impact of circuit diversity on labeled training data (DF set [25]). We used DF as labeled data and AWF2 as unlabeled data. All standard deviations are less than 0.5%.

train (25)	acc	train (90)	acc
1 circuit	86.6	slow	93.4
5 circuits	86.8	fast	92.9
40 circuits	87.1	random90	93.5

performance for each technique is shown in Table 3. The best results for a given number of training instances is *shown in bold.*

Our experiments show that GANDaLF is effectively tied with TF when using 10 samples per class. However, the testing cost of GANDaLF was lower than TF, DF, and Var-CNN. For 50 samples and above, Var-CNN is the best classifier, but it was much less effective when limited to 5 or 10 samples, with accuracies of 26% and 69%, respectively. In the lowest data setting with 5 samples, TF was the most accurate classifier due to its pre-trained WF model. Across all classifiers, if the attacker can afford this larger cost for data collection, the payoff is worthwhile for closed-world classification of index pages. In particular, when either DF or Var-CNN is trained on many more instances, performance is much improved. When trained on 90 instances, the accuracy of DF and Var-CNN improves to 97% and 98% respectively. This is important evidence to suggest that these models require very large labeled training datasets to learn effective feature representations in the WF-I scenario.

Impact of Circuit Diversity. The attacker using GANDaLF needs to gather and use a smaller dataset of labeled data than in other attacks, so the source of that data may impact attack accuracy. In particular, the circuits used to collect this data might be slow, fast, or otherwise not representative of the kinds of conditions faced by the victim. To investigate the impact on GANDaLF of the diversity of circuits used to gather labeled training data, we examine how the number of circuits used to gather data impacts accuracy. We used a subset of the DF dataset collected using 40 circuits, and split it into 40 smaller subsets, one per circuit. Each subset consists of 25 instances of each of 95 websites. Thus, we randomly sampled 95 websites and 25 instances (95×25) within one subset, four subsets, and 40 subsets to construct three training labeled sets and 100 instances of each of 95 websites within all 40 subsets to build one testing set (We detailed this data sampling in Algorithm 2 of Appendix C). Then we trained three different models using each labeled set and tested them using the testing set. As shown in Table 4, the performance somewhat improved with increasing number of circuits, though it is far from critical in performing the attack.

Impact of Network Conditions. While the attacker would likely use multiple circuits to gather labeled training data, the victim may have a particularly slow or fast circuit. Thus, we examine how the speed of the victim's circuit impacts the attack. We use the same 40 subsets of the DF dataset as when testing circuit diversity. We split out the four fastest circuits and the four slowest circuits by using the total website load times.

We then constructed fast (or slow) testing sets by randomly sampling 95×100 instances from data gathered using the four fast circuits (or slow circuits), which was the same testing set size used in DF [25]. To train GANDaLF, we randomly chose 95×90 instances over the remaining 36 subsets. We described this data sampling details in Algorithm 3 of Appendix C.

As a baseline, we further trained GANDaLF using randomly chosen 95×90 samples over 40 subsets and tested it using another randomly chosen 95×100 instances over 40 subsets. As shown in Table 4, GANDaLF performs modestly worse when identifying traces using fast circuits and about the same on slow circuits. The small margins indicate that the network condition when collecting the victim's traffic minimally impacts the performance of GANDaLF.

Impact of Unlabeled Data. To understand the impact of the choice of unlabeled data, we also trained GANDaLF using the AWF1 set as both labeled and unlabeled data. In other words, this models the case that both sample groups come from the same distribution. Interestingly, this change only led to a 1% increase in the accuracy of CW classification. We further trained GANDaLF using the DF set (which was collected in different network settings) and GANDaLF yielded 87% CW accuracy when using 20 labeled training instances per website. Even though the distributions of distances between labeled and unlabeled sets were somewhat different as shown in Figure 3, this result shows that the gap did not critically impact the classification ability of GANDaLF. This suggests that the unlabeled data does not require either any of the monitored sites in the labeled set or the same network setting for the unlabeled data collection to provide a useful basis for semi-supervised learning.

OW Performance. Since GANDaLF and TF performed more effectively in the low-data CW setting (i.e.,

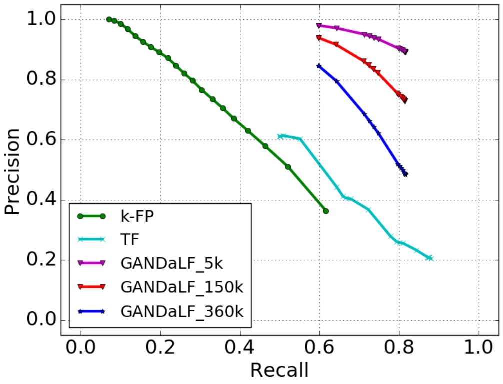


Fig. 4. WF-I, OW: Comparison to k-FP and TF. We used 360k background traces for k-FP and TF.

5-10 instances), we further evaluated them in the open-world scenario. In this evaluation, the classifiers were trained using 20 instances for each monitored site in AWF1 and 2,000 unmonitored site instances from AWF-OW. We then tested using a background set of 360,000 unmonitored website samples, which is the same size as the largest background set explored by TF in their OW evaluations [26]. We further varied the size of the unmonitored set from 5,000 to 360,000 to show the impact of the background set on the performance of GANDaLF. As shown in Figure 4, GANDaLF outperformed k-FP and TF and increasing the unmonitored set size degraded GANDaLF performance. Compared to the CW setting, GANDaLF provides better performance than TF by a more significant margin in detecting monitored websites versus unmonitored websites. The better effectiveness in OW scenarios is mainly because the discriminator using additional supervised loss also became more benefited by the binary classification setting.

6.3 Fingerprinting Websites with Subpages

In this section, we investigate the classification ability of GANDaLF and other techniques in the WF-S setting. This scenario not only represents a more realistic scenario for attacks, but also a more challenging one, as the inclusion of many subpage traffic instances results in high intra-class variation.

CW Performance. For the WF-S CW experiments, we trained each technique in a low-data setting with between 5-90 training instances per site, where each instance was randomly sampled. This means that, at best, the attacker is able to train on one sample per subpage within each site in our dataset. Consequently, during experiments where the training sample count is below 90, there are subpages within the testing set on which the attack did not train. We believe this challenging CW scenario appropriately models the real-world difficulty of accurately profiling an entire website under reasonable data restrictions.

Table 5. WF-S, CW: Comparison to k-FP, DF, Var-CNN (Var), and TF using 5-90 training instances. For unlabeled sets, we used AWF1 ($\mathbf{GF(A)}$) or GDLF-OW-old ($\mathbf{GF(G)}$). We do not show standard deviations less than 1%. We measured the time (s: seconds) for testing 12k testing samples. Other numbers are %.

TrainN	GF(A)	GF(G)	k-FP	DF	Var	TF
5	30±1	31±2	41	4	5±1	14±1
10	39±3	38±2	46	5±1	6±2	17±1
20	46±3	47±3	52	47±2	9±2	18
50	57±2	56±3	57	49±2	21±6	18
90	62±1	62±3	61	61	25±7	19
time	2.3s	2.3s	3.1s	5.1s	12.07s	8.2s

As shown in Table 5, this difficult training scenario and the higher intra-class variance reduced the performance for all WF methods. GANDaLF performed the most effectively using 90 instances per site and ties with k-FP when using 50 instances per site. In the lower data settings, however, k-FP is more accurate. We believe that the *categorical* features such as the total number of packets enabled k-FP to gain an enhanced understanding about subpage traces even using more limited training data. In contrast, deep learning models must learn feature representations from scratch using the few training samples provided, inevitably resulting in the network gaining a poorer understanding of the data.

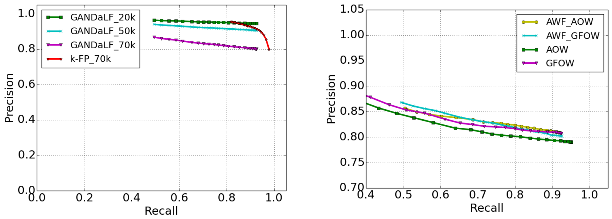
TF and Var-CNN achieved worse performance than anticipated for all cases. The poor performance of Var-CNN may be explained by how heavily tuned the model is to the traditional WF-I scenario. The expanded receptive field of the dilated convolutions used by the network may cause the model to miss meaningful local patterns.

For TF, it seems that the distinctions between AWF websites did not help the model generate good features for the subpage traces, because the decision boundary for the classification in WF-S was different from WF-I. Since the pre-trained model was trained using labels, the decision boundary became more biased towards WF-I, leading to poor feature embeddings for subpage traces.

In contrast, GANDaLF was trained by additional unsupervised loss and feature matching loss, enabling it to learn a broader view of AWF1 traces without labels rather than focusing on the differentiation between AWF websites based on labels. This makes GANDaLF

Table 6. GANDaLF CW accuracy (**Acc**) according to different labeled set by varying the number of subpages (s) and instances (i) in the labeled set. All numbers are %.

$s \times i$	2×10	10×2	20×1	random
Acc	**42.78±2.3**	**46.03±3.5**	**45.91±2.0**	**46.7±2.0**



(a) Comparison to k-FP. **(b)** Impact of unlabeled set.

Fig. 5. WF-S, OW: GANDaLF OW experiment by varying the background sizes and unlabeled sets.

Table 7. Various unlabeled data settings using AWF1, AWF-OW (**AOW**), and GDLF-OW-old (**GOW**). We reported the trace count (**size**), whether or not the network setting was different from the GDLF25 setting (**network**), and time gap (**y**: years, and **m**: months).

setup	AWF1-AOW	AWF1-GOW	AOW	GOW
size	**649k**	**329k**	**400k**	**80k**
network	**no-no**	**no-yes**	**no**	**yes**
timegap	**3y-3y**	**3y-3m**	**3y**	**3m**

performance solid even when learning from a different distribution (WF-S versus WF-S-A1 in Figure 3).

Impact of Subpages in Labeled Data. We further varied the number of subpages used to represent each website in the training data. More specifically, we varied the number of subpage classes by fixing the total instance count at 20. For example, we varied the subpage count s and instance count i to be $s \times i = 20$. We first randomly selected s subpages among 96 subpages and then randomly sampled i instances for each subpage. This scenario allows us to better see the effects of the increased intra-class variance as the number of subpages (i.e., s) increases to labeled training data and further guides us to build the optimal labeled set to maximize the GANDaLF performance.

Based on Table 6, building the labeled set by random sampling without considering s performed slightly better than other cases while GANDaLF performance remarkably worsened with more limited subpages when s=2. This indicates that enough variance between subpages in the labeled set is important to maximize the performance of GANDaLF.

Impact of Unlabeled Data. As briefly discussed in Section 5.3, we studied the effect of unlabeled sets to the classification performance as well as the generator loss. In this experiment, we used two unlabeled sets, AWF1 and GDLF-OW-old which has a three-month time gap with GDLF25 set, and one labeled set from GDLF25. The generator loss somewhat decreased with GDLF-OW-old while it kept increasing with AWF1. Even though the GDLF-OW-old set made training more stable, the CW accuracy was comparable across most settings based on Table 5 (GF(A) versus GF(G)). However, the use of unlabeled set built from different data distribution degraded the generator ability, which led to more biased training towards a better discriminator and may eventually result in limiting the upper-bound of supervised learning capacity since L_u in Algorithm 1 of Appendix B also hardly decreased.

To create good fake samples against a greater number of classes than examined by Saliman et al. [23], we should feed a much larger unlabeled set of subpage traces in which the corpus of the websites, subpages, and instances is tremendous to train the generator effectively. As a result, both generator and discriminator may reach the optimal Nash equilibrium while gaining powerful supervised performance with a high CW accuracy. We leave further investigation on the usage of more optimal unlabeled data as future work.

OW Performance. We further conducted an OW evaluation of GANDaLF and k-FP in the WF-S setting, since they had better performance than other WF attacks in the CW evaluation. For this experiment, we trained both classifiers using the labeled set consisting of 90 instances of 25 monitored sites and 2,250 unmonitored subpages. In addition, we used AWF1 and AWF-OW sets as unlabeled data for GANDaLF. Figure 5a shows that as we increase the size of the unmonitored set, GANDaLF becomes less effective, as expected. In particular, k-FP outperformed GANDaLF in the OW setting, which indicates that the handcrafted features provide a more consistent basis to identify pages from sites in the monitored set than the GANDaLF model.

We further investigated how combining unlabeled sets could amplify the performance of GANDaLF. For this experiment, we adopted AWF1, AWF-OW, and GDLF-OW-old. We created combined datasets of AWF1 with AWF-OW and AWF1 with GDLF-OW-old, and compared these against AWF-OW and GDLF-OW-old by themselves. See Table 7 for details.

Figure 5b shows that both of the combined unlabeled sets performed slightly more effectively. This suggests that the amount and perhaps variety of unlabeled data played a role in enhancing the performance of GANDaLF, even though some of the data was collected three years prior to the labeled data and from different network conditions (i.e., AWF-AOW in Table 7). Furthermore, GANDaLF improved with the inclusion of the GDLF-OW-old set, which suggests that the unlabeled subpage traces help generate good fake samples to distinguish monitored subpages from unmonitored subpages by lowering L_u and L_G in Algorithm 1 of Appendix B.

Summary. We find that GANDaLF outperforms other DL-based classifiers on subpages. Surprisingly, however, k-FP was even more effective in both the CW and OW settings. The greater intra-class variation made it harder for automatic feature extraction to work effectively, while manually defined features can still work consistently in such a challenging setting.

7 Conclusion

We introduce a novel attack, GANDaLF, using GANs in the semi-supervised setting, in which the generator minimizes the difference between real trace and fake trace distribution while the discriminator is trained to distinguish between real and fake samples and, further, improve classification over the labeled set, by leveraging both labeled and unlabeled traces. Because it requires only a small amount of labeled data, we investigated the applicability of this variant of GANs in the low-data setting for WF attacks. Furthermore, we evaluated GANDaLF by considering both sites' index and non-index pages using various experimental scenarios. Finally, our empirical study showed that GANDaLF had better performance than Var-CNN and TF, the most recent low-data WF attacks, at non-index fingerprinting, with particularly significant performance advantage in the open-world setting. However, in WF-S, GANDaLF did not become more effective than k-FP leveraging the total packet statistics.

Reproducibility. The source code and datasets used in this paper are available on Github.[2]

Acknowledgments

We thank the anonymous reviewers for their many comments that helped to improve the paper. This work was funded in part by the National Science Foundation under Grants nos. 1722743, 1816851, 1433736, and 1815757.

References

[1] Code for the paper "improved techniques for training GANs. https://github.com/openai/improved-gan.

[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-.

[3] Tor browser crawler. https://github.com/webfp/tor-browser-crawler.

[4] S. Bhat, D. Lu, A. Kwon, and S. Devadas. Var-CNN: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(4):292–310, 2019.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[6] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security Symposium*, pages 1187–1203, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *ACM workshop on Cloud computing security*, pages 31–42, 2009.

[9] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6626–6637, 2017.

[10] R. Jansen, M. Juarez, R. Galvez, T. Elahi, and C. Diaz. Inside job: Applying traffic analysis to measure Tor from within. In *Network & Distributed System Security Symposium (NDSS)*, 2018.

[11] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *ACM Conference on Computer and Communications Security (CCS)*, pages 263–274. ACM, 2014.

[12] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security (ESORICS)*, pages 27–46. Springer, 2016.

[13] B. Lecouat, C.-S. Foo, H. Zenati, and V. R. Chandrasekhar. Semi-supervised learning with GANs: Revisiting manifold regularization. *arXiv preprint arXiv:1805.08957*, 2018.

[2] https://github.com/traffic-analysis/gandalf

[14] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr. Understanding Tor usage with privacy-preserving measurement. In *Internet Measurement Conference*, pages 175–187, 2018.

[15] S. E. Oh, S. Li, and N. Hopper. Fingerprinting keywords in search queries over Tor. *Proceedings on Privacy Enhancing Technologies*, 2017(4):171–190.

[16] S. E. Oh, S. Sunkam, and N. Hopper. p1-FP: Extraction, classification, and prediction of website fingerprints with deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(3):191–209, 2019.

[17] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at Internet scale. In *Network & Distributed System Security Symposium (NDSS)*, 2016.

[18] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2011.

[19] M. Perry. Padding negotiation. Tor Protocol Specification Proposal. https://gitweb.torproject.org/torspec.git/tree/proposals/254-padding-negotiation.txt, 2015.

[20] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright. Tik-Tok: The utility of packet timing in website fingerprinting attacks. *Proceedings on Privacy Enhancing Technologies*, 2020(3):5–24, 2020.

[22] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen. Automated website fingerprinting through deep learning. In *Network & Distributed System Security Symposium (NDSS)*, 2018.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2234–2242, 2016.

[24] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 901–909, 2016.

[25] P. Sirinam, M. Imani, M. Juarez, and M. Wright. Deep Fingerprinting: Undermining website fingerprinting defenses with deep learning. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2018.

[26] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright. Triplet Fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1131–1148, 2019.

[27] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[28] I. Sutskever, R. Jozefowicz, K. Gregor, D. Rezende, T. Lillicrap, and O. Vinyals. Towards principled unsupervised learning. *arXiv preprint arXiv:1511.06440*, 2015.

[29] J. van de Wolfshaar. Semi-supervised learning with GANs. Medium Blog. https://medium.com/@jos.vandewolfshaar/semi-supervised-learning-with-gans-23255865d0a4, 2018.

[30] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security Symposium*, pages 143–157, 2014.

[31] T. Wang and I. Goldberg. Walkie-Talkie: An efficient defense against passive website fingerprinting attacks. In *USENIX Security Symposium*, 2017.

Appendix

A Discriminator Loss Implementation

We implemented the discriminator and generator training based on an existing implementation of SGAN by Wolfshaar, which is described in a blog post [29]. In this section, we briefly discuss the details of the discriminator loss implementation based on the interpretation given in the blog post [29].

As Saliman *et al.* [23] suggested that having $K+1$ softmax distribution is overparameterized, subtracting a function $f(x)$ from each output logit hardly impacts the output of the softmax. Thus, this is equivalent to fix $l_{K+1}(\mathrm{x}) = 0 \forall \mathrm{x}$ where the supervised loss becomes the standard supervised loss function of the classifier with K classes and the discriminator, D, will be $D(x) = \dfrac{Z(x)}{Z(x)+1}$ where $Z(x) = \sum_{k=1}^{K} \mathsf{exp}(l_k(\mathrm{x}))$. As such, $Z(x)$ is the sum of the unnormalized probabilities. Since we want to take the log probability of the fake class for our loss function,

$$\begin{aligned} & log(Z(x)) - log(1 + (Z(x))) \\ & = \mathsf{logsumexp}(l_1, ...l_K) - \mathsf{softplus}(\mathsf{logsumexp}(l_1, ...l_K)), \end{aligned}$$

where l: the output logits, K: the number of classes in the labeled set, and $\mathsf{softplus}(\mathrm{x}) = log(1+x)$.

Since the generative adversarial training requires to ascend the gradients of $log(D(x)) + log(1\text{-}D(G(z)))$ (i.e., vanilla GAN loss function), the optimizer achieves the following,

$$\begin{aligned} & -log(D(x)) - log(1 - D(G(z))) \\ & = -log(Z(x)/(1 + Z(x))) \\ & - log(1 - Z(G(x)))/(1 + Z(G(z))) \\ & = -log(Z(x)/(1 + Z(x))) - log(1/(1 + Z(G(z)))) \end{aligned}$$

By using the softplus function, the unsupervised loss of the discriminator is implemented as follows.

$$\begin{aligned}&\mathsf{softplus}(\mathsf{logsumexp}(l_1^x, ..., l_K^x)) - \mathsf{logsumexp}(l_1^x, ..., l_K^x)\\ &+ \mathsf{softplus}(\mathsf{logsumexp}(l_1^{G(z)}, ..., l_K^{G(z)}))\end{aligned}$$

Saliman *et al.* [1, 23] implemented this for CIFAR-10 dataset and we also used it for GANDaLF.

B GANDaLF Training

We present the pseudocode of training GANDaLF to detail our experimental setup in Algorithm 1.

Algorithm 1: GANDaLF training.

```
Input  : Labeled examples $(x_l,\ y_l) \sim p_{d_1}$, Unlabeled
         examples $(x_u) \sim p_{d_2}$, latent variable
         $z \sim p(z)$, number of iterations $i$, $\alpha_1$=2$e^{-4}$,
         $\alpha_2$=5$e^{-5}$, and $\beta$=0.5.
$G$   – generator network
$D$   – discriminator network
$f$   – output of the flatten layer of D
$L_D$ – discriminator loss
$L_G$ – generator loss
$\omega$   – parameters of discriminator
$\theta$   – parameters of generator
for $i$ = 1 to $m$ do
    $\tilde{x} \leftarrow G(z)$
    $\hat{x_z},\ f_z \leftarrow D(\tilde{x})$
    $\hat{x_l},\ f_l \leftarrow D(x_l)$
    $\hat{x_u},\ f_u \leftarrow D(x_u)$
    $L_s \leftarrow$ CrossEntropy($\hat{x_l},\ y_l$)
    $L_u \leftarrow$ - logsumexp($\hat{x_u}$) + softplus(logsumexp($\hat{x_u}$))
      + softplus(logsumexp($\hat{x_z}$))
    $L_D \leftarrow L_s + L_u$  /* supervised+unsupervised loss */
    $\omega \leftarrow$ Adam($\nabla_\omega \frac{1}{m} \sum L_D^{(i)}, \omega, \alpha_2$)  /* D optimizer */
    $L_G \leftarrow$ MAE($f_z,\ f_u$)                     /* MSE in WF-S */
    $\theta \leftarrow$ Adam($\nabla_\theta \frac{1}{m} \sum L_G^{(i)}, \theta, \alpha_1, \beta$)    /* G optimizer
      */
end
```

C Data Sampling for the Experiments to Show the Impact of the Network Condition

To show the effect of the network condition to GANDaLF performance, we studied two scenarios by varying the circuit diversity involved in the data collection and the network congestion when collecting the testing set. In this section, we detailed how we constructed training and testing sets to show the impact of circuit diversity (Algorithm 2) as well as network congestion (Algorithm 3).

Algorithm 2: Data sampling to generate labeled sets by varying the circuit diversity.

```
Input  : DF dataset ($D = (X, Y_l, Y_c)$), total circuit
         index array ($C = \{1, 2, ..., 40\}$), circuit count
         ($n_c$), website count ($n_w$), labeled sample
         count per class ($n_l$), and testing sample
         count per class ($n_t$).
Output: Training data ($I_{tr}$) and testing data ($I_{te}$).
Shuffle $D$.        /* (samples, labels, circuit labels) */
Shuffle $C$.
Initialize $C_{tr}$, $C_{te}$,$D_{tr}$, $D_{te}$, $I_{tr}$, $I_{te}$.
$C_{tr} \leftarrow$ randomly chosen $n_c$ entries in $C$.
for ($x, y_l, y_c$) in $D$ do
    if $y_c$ in $C_{tr}$ then
        $D_{tr} \leftarrow D_{tr} \cup (x, y_l)$
    end
    if $n_c < 36$ then
        /* To ensure that circuits in $C_{te}$ should have
           at least 9,500 entries since each circuit
           subset consists of 95×25. */
        $C_{te} \leftarrow \{C - C_{tr}\}$.
        if $y_c$ in $C_{te}$ then
            $D_{te} \leftarrow D_{te} \cup (x, y_l)$
        end
    else
        $D_{te} \leftarrow$ randomly chosen $n_w \times n_t$ entries in
          $\{D - D_{tr}\}$
    end
end
                    /* sampling for each website subset. */
for $i$ in ($\{1, 2, ..., n_w\}$) do
    $I_{tr} \leftarrow$ randomly chosen $n_l$ instances in $D^i_{tr}$
    $I_{te} \leftarrow$ randomly chosen $n_t$ instances in $D^i_{te}$
end
```

Algorithm 3: Data sampling to simulate the victims with fast or slow circuits.

```
Input  : DF dataset ($D = (X, Y_l, Y_c)$), total circuit
         index array ($C = \{(1, t_1), ..., (40, t_{40})\}$),
         website count ($n_w$), labeled sample count
         per class ($n_l$), and testing sample count per
         class ($n_t$).
Output: Training data ($I_{tr}$) and testing data ($I_{te}$).

Shuffle $D$.        /* (samples, labels, circuit labels) */
Shuffle $C$.        /* (circuit index, mean of site loading
  time). */
$C_{index} \leftarrow \{1, 2, ..., 40\}$                /* index. */
$C_{time} \leftarrow \{t_1, t_2, ..., t_{40}\}$          /* mean loading time. */
Initialize $C_f$ with top 4 indices in reverse($C_{time}$).
Initialize $C_s$ with top 4 indices in $C_{time}$.
Initialize $C_{tr}$, $C_{te}$,$D_{tr}$, $D_{te}$, $I_{tr}$, $I_{te}$.
if $choice$ == "fast" then
    $C_{te} \leftarrow C_f$  /* use fast subsets as testing data. */

else
    $C_{te} \leftarrow C_s$  /* use slow subsets as testing data. */

end
$C_{tr} \leftarrow \{C_{index} - C_{te}\}$
for $(x, y_l, y_c)$ in $D$ do
    if $y_c$ in $C_{tr}$ then
        $D_{tr} \leftarrow D_{tr} \cup (x, y_l)$
    end
    if $y_c$ in $C_{te}$ then
        $D_{te} \leftarrow D_{te} \cup (x, y_l)$
    end
end
                    /* sampling for each website subset. */
for $i$ in $(\{1, 2, ..., n_w\})$ do
    $I_{tr} \leftarrow$ randomly chosen $n_l$ instances in $D^i_{tr}$
    $I_{te} \leftarrow$ randomly chosen $n_t$ instances in $D^i_{te}$
end
```