mathematics

MDPI

Article

A Practical Website Fingerprinting Attack via CNN-Based Transfer Learning

Tianyao Pan [1], Zejia Tang [2] and Dawei Xu [3,*]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China; tianyaopan@bit.edu.cn
[2] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China; 3120211369@bit.edu.cn
[3] College of Cybersecurity, Changchun University, Changchun 130022, China
* Correspondence: xudw@ccu.edu.cn

Abstract: Website fingerprinting attacks attempt to apply deep learning technology to identify websites corresponding to encrypted traffic data. Unfortunately, to the best of our knowledge, once the total number of encrypted traffic data becomes insufficient, the identification accuracy in most existing works will drop dramatically. This phenomenon grows worse because the statistical features of the encrypted traffic data are not always stable but irregularly varying in different time periods. Even a deep learning model requires good performance to capture the statistical features, its accuracy usually diminishes in a short period of time because the changes of the statistical features technically put the training and testing data into two non-identical distributions. In this paper, we first propose a convolutional neural network-based website fingerprinting attack (CWFA) scheme. This scheme integrates packet direction with the timing sequence from the encrypted traffic data to improve the accuracy of analysis as much as possible on few data samples. We then design a new fine-tuning mechanism for the CWFA (FM-CWFA) scheme based on transfer learning. This mechanism enables the proposed FM-CWFA scheme to support the changes in the statistical patterns. The experimental results in closed-world and open-world settings show that the effectiveness of the CWFA scheme is better than previous researches, with the slowest performance degradation when the number of data decreases, and the FM-CWFA scheme can remain effective when the statistical features change.

Keywords: website fingerprinting attack; convolutional neural network; fine-tuning mechanism; transfer learning

MSC: 68T07

Citation: Pan, T.; Tang, Z.; Xu, D. A Practical Website Fingerprinting Attack via CNN-Based Transfer Learning. *Mathematics* **2023**, *11*, 4078. https://doi.org/10.3390/math11194078

Academic Editors: Jialing He, Zhi Fang, Chunhai Li and Antanas Cenys

Received: 1 September 2023
Revised: 17 September 2023
Accepted: 22 September 2023
Published: 26 September 2023

1. Introduction

Due to an increasing awareness of privacy protection, internet users tend to use anonymous tools, such as the onion router (Tor, for short), when communicating to achieve anonymous access. Tor [1] is one of the largest anonymous networks worldwide, with over six thousand relay nodes and approximately three million daily users. When using Tor, users randomly select three relay nodes. These nodes provide the function of link encryption for cutting down the relationship between a user's IP address and the target website. Thanks to the link encryption, a user's visiting website via Tor cannot be traced directly, even if any one of the relay nodes is compromised by the adversary.

However, only adopting link encryption is insufficient for resisting traffic analysis attacks because AI models can learn the pattern of a user's visit. A website fingerprinting (WF) attack is such a kind of traffic analysis attack. Specifically, a WF attacker first eavesdrops the encrypted traffic by corrupting a relay node. Then, the pattern is defined by the specific features in the traffic. Finally, the WF attacker feeds these features into an artificial intelligence (AI) classifier to infer the website that is visited by a user.

There are two classical types of WF attacks from the point of view of the AI classifier [2–4]. Some works built the classifier via traditional machine learning algorithms, while the others proposed the classifier via deep learning algorithms. The former extracts features that represents traffic (e.g., the timestamp in the packet data) in manual way and then invokes classifier algorithms (e.g., the support vector machine or random forest) to execute the task of classification. On the contrary, the later directly relies on the training data themselves to train the classifier by deep neural networks (e.g., convolutional neural networks or transformers).

With the rapid development of deep learning, the second type of WF attack has gradually become a hotspot. Several new deep neural networks [5–7] have been proposed in recent years. The advantage of these attacks is that feature extraction and classification are integrated into one model naturally, so empirical-based features are not required. In other words, all the features are automatically extracted through the neural networks. This makes them more likely to discover deeper intrinsic features from the trace of the encrypted traffic, resulting in exceptional accuracy. However, deep learning methods require a large number of training data, which makes them less practical than the traditional classifiers when the number of data is not sufficiently large. Moreover, training neural networks usually takes a large amount of time.

What is worse, the content of a website may incessantly change in reality, which causes the pattern to vary in encrypted traffic from time to time. This is known as concept drift. Prior works [8] demonstrated that concept drift can seriously affect the performance of the WF attacks, so the attacker needs to regularly update the data and retrain the AI classifier. To deal with this challenge, some recent works [9–11] have started to adopt few-shot learning and transfer learning to reduce the effort of collecting data and training classifiers. But the accuracy of most works is too low to be used in reality. Therefore, knowing how to balance accuracy and training overhead is a technical challenge that needs to be addressed in deep learning-based WF attacks.

Meanwhile, several defense strategies [12–17] were proposed to fight the WF attacks. Some strategies show the feasibility of practical deployment in Tor due to their reasonable cost and effectiveness. However, to the best of our knowledge, current attacks cannot maintain high accuracy anymore because the traffic traces are protected by website fingerprinting defense strategies.

In this work, we investigate whether deep learning can be used to achieve good classification results with small datasets and how to guard against concept drift. In addition, we explore the effects of package direction and timing traces on classification. The main contributions of this paper are summarized as follows:

- We first propose a new convolutional neural network-based website fingerprinting attack (CWFA) scheme. It integrates both package direction and time sequence information. It can alleviate the decline of accuracy and resist common defense strategies, when compared with existing attacks.
- We then design a fine-tuning mechanism for the proposed CWFA scheme. The proposed FM-CWFA scheme can tackle the irregular changes of the statistical features.
- Systematic experiments show that the decline of the performance in the CWFA scheme is the slowest due to the decreasing of the training data, while the FM-CWFA scheme is still valid even if the training and testing data are collected three years apart.

The rest of the paper is organized as follows. The related works are recalled in Section 2. Section 3 briefly introduces the preliminaries, and the background of WF attacks, including the threat model and related assumptions, is presented in Section 4. In Section 6, we introduce the methodology. Then, Sections 6.2 and 6.3 present the design of the scheme, and Section 7 provides a comprehensive comparison of the proposed scheme with recent related works. Finally, the possible future works are discussed in Section 8, and the paper is concluded in Section 9.

2. Related Works

In this section, we categorize and describe previous works on the exploration of the WF attacks and defenses.

2.1. The WF Attack

The original attempt to attack Tor was first considered by Herrmann et al. [18] in 2009. However, the accuracy was only 3% in their scheme. From then on, researchers start to use various features to bridge the gap between a user's IP address with the target website. For an instance, the length of every IP packet is usually an important dimension of describing the features. Therefore, due to the fixed size of the data packets sent by Tor (512 bytes), the previous identification models are invalid for the classification on Tor traffic. In 2011, Panchenko et al. [19] analyzed Tor traffic features and used support vector machines (SVM) as classifiers to improve the accuracy in the closed-world. More precisely, the accuracy of their scheme varies from 3% to 55%. Since then, people tried to propose more powerful classifiers and design more effective features to improve the accuracy of identifying the websites. Wang et al. [2] extracted features such as packet ordering, outgoing packet concentration, and burst and used a k-nearest neighbor (k-NN) classifier for classification. Subsequently, Panchenko et al. [3] proposed the CUMUL attack, which is an other SVM classifier-based attack that mainly relies on cumulative packet length features. Hayes et al. [4] proposed the k-fingerprinting (k-FP) attack. Unlike k-NN and CUMUL, this attack used random forest (RF) and feeds the extracted feature vectors into the k-NN classifier for classification. They believed that using the extracted new features was more effective than using the traditional original features. The accuracy of these works can reach up to 90%.

However, manual feature extraction was too empirical, difficult, and expensive. Moreover, they were bound to be limited by feature sets. Once defenses or protocols were changed, these features could be hidden, rendering the attack ineffective immediately.

Due to the effectiveness of deep learning techniques in the fields of image recognition and natural language processing, researchers discovered their potential in the field of encrypted traffic analysis. Many recent studies [5–7,20] have adopted deep learning techniques for WF attacks and demonstrated their effectiveness in the field of traffic identification. Compared with traditional attacks, deep learning-based attacks do not require hand-crafted features as they are able to automatically extract features from the original sequence. These attacks are gradually becoming the mainstream method of WF attacks. We select several deep learning techniques to compare with our attack.

Deep Fingerprinting: Sirinam et al. [7] proposed deep fingerprinting (DF) in 2018; they borrowed the idea of the VGG [21] model and designed a more complex CNN architecture. It uses only direction sequence and achieves 98% accuracy in the closed-world setting when trained on a dataset of 95 websites with 1000 traces per website. Furthermore, DF is the first attack to effectively undermine WTF-PAD [12] with over 90% accuracy. However, DF uses a large number of data for training, and the accuracy will drop rapidly when the training samples are reduced, making it unable to meet existing requirements.

Var-CNN: Bhat et al. [20] designed Var-CNN, and they found that in addition to direction information, it is also effective to use timing information for WF attacks. In order to apply direction and timing information simultaneously, they combined direction sequences and timing sequences with a small number of hand-crafted features, respectively, trained two CNN models based on ResNet [22], and integrated the two models to achieve higher accuracy. They claimed that this attack can achieve good results on smaller datasets, but it still does not get rid of the dependence on hand-crafted features. Moreover, since Var-CNN needs to train two models and the model structure is complex, the time cost of training is high.

Triplet Fingerprinting: Sirinam et al. [9] proposed triplet fingerprinting (TF), which leverages the idea of N-shot learning. They used triplet network as feature extractor and k-NN classifier as target classifier, allowing the attacker to train only a few samples on

each target website. TF is able to achieve 85% accuracy with only 5 samples per website when the data used for training and testing is collected separately over the years and on different networks. However, TF becomes ineffective when traffic is protected by WF defense schemes.

Adaptive Fingerprinting: Wang et al. [10] proposed adaptive fingerprinting (AF), which leverages the method of adversarial domain adaptation. It also needs to pre-train the feature extractor, extract the embedded features of the target dataset, and finally feed these features to the k-NN classifier for website fingerprinting, but it can use multiple source datasets to train the feature extractor. An AF attack performs better than TF and takes less time to train the feature extractor, but it does not show advantages when the sample size is small.

Transfer Learning Fingerprinting Attack: Chen et al. [23] proposed the transfer learning fingerprinting attack (TLFA); they used training data collected from non-target websites to train a powerful embedding model and then used a small set of labeled training data from target websites to fine-tune a task-specific classifier model. However, they did not propose a more effective WF attack model but only gave a model fine-tuning method for the few-shot WF attack problem. And this method assumes that the pre-training dataset and the target dataset have similar data distributions, without considering the more difficult setting of different data distribution patterns.

2.2. The WF Defense

To defend against local passive attackers, researchers have proposed many defenses against WF attacks. Since attackers mainly use features in traffic traces to identify traffic, the purpose of WF defenses is often to hide traffic features and make them less identifiable. Many studies hide traffic features by fixing the packet transmission rate, such as buffered fixed-length obfuscation (BuFLO) [14], congestion-sensitive BuFLO (CS-BuFLO) [16], and Tamaraw [15]. These defenses are effective against nearly all known attacks, but their large overhead prevents them from actually being deployed in Tor. There are also some defenses that disrupt website fingerprints by randomly injecting dummy packets or artificial delays, which tend to have less latency and bandwidth overhead.

In this paper, we use two lightweight defenses to evaluate the impact of defenses on WF attacks: WTF-PAD and FRONT.

WTF-PAD: Juarez et al. [12] proposed a lightweight defense WTF-PAD using an adaptive padding technique. WTF-PAD detects the delay time between consecutive bursts and adds dummy packets when the delay time is large. Since the defense does not add packet delays, it has no latency overhead, and it can reduce the accuracy of k-NN to below 20% with 54% bandwidth overhead; however, the DF attack can still achieve 90% accuracy.

FRONT: FRONT [13], proposed by Gong and Wang, is also a lightweight defense strategy. Considering that the front part of the traffic trace contains rich features, FRONT focuses on obfuscating the trace with dummy packets. Apart from that, the defense adds dummy packets in a highly random manner to ensure that different traces of the same website have different features from each other. FRONT is effective against most attacks, including DF, with a similar overhead to WTF-PAD.

2.3. Website Fingerprint Selection

In early WF attacks, some use low-level timing information as features, such as total transmission time, and inter-packet timing. But these low-level timing features do not provide much contribution to the classifier. The study of the importance of features by Hayes et al. [4] shows that the usefulness of these low-level timing features for classification can be ignored. Influenced by this study, the WF attacks using deep learning in recent years often only use the direction of packets as input, such as AWF [6] and DF [7]. At the same time, many WF defenses only consider obfuscation of packet direction sequences. These defenses only add dummy packets to the trace but do not perform any time delays.

However, Bhat et al. [20] conducted experiments on packet timing. They found that a model using inter-packet timing information has almost comparable accuracy to that using direction information. Rahman et al. [24] also proved that the value of timing information for WF was seriously underestimated.

3. Preliminaries

3.1. The CNN Model

We use CNN to build the deep neural networks. CNN is a deep network widely used in classification tasks and has proven its effectiveness in image classification, speech recognition, and other fields. It mainly automatically extracts features from the original input data through multiple convolutional layers, pooling layers, and nonlinear activation functions (e.g., ReLu). Batch normalization (BN) and dropout layers are usually used after the convolutional layer to prevent overfitting and improve performance. The last part of CNN is the full connectioned (FC) layer, which combines all local features into global features to calculate the final score of each category.

Two-dimensional convolution (2D-CNN) is usually used in the field of image classification, but WF attacks need to pay attention to the sequence information of the trace, which is difficult for 2D-CNN. One-dimensional convolution (1D-CNN) is mainly used to extract features from time-series data with only one dimension. It can extract the sequence information of one-dimensional data; therefore, it is more appropriate to use 1D-CNN for WF attacks.

3.2. Transfer Learning

Website fingerprinting attacks face the problem of concept drift, and it is difficult to repeatedly collect new data. To solve this problem, it is natural to think of using transfer learning.

Transfer learning is a machine learning technique that can transfer the knowledge learned on the source task to the target task, thereby improving the performance of the target task model prediction. The reason why transfer learning is effective is that the first layers of the model learn general features, and with the deepening of the network, the latter network is more focused on learning specific features. This makes it possible to directly transfer the early layers and then adjust the deeper layers to adapt to the new task.

Fine-tuning is a method of transfer learning, which saves a lot of computing resources and time. If the new dataset is similar to the pre-trained dataset, then fine-tuning on the trained model can adapt the model to the new dataset.

3.3. Loss Function

Cross-entropy loss function is often used for multi-classification tasks; the traditional cross-entropy loss function is as follows:

$$\begin{aligned} L_{normal} &= -\sum_{i=1}^{K} y_i log(p_i), \\ y_i &= \begin{cases} 1, & i = true\,label \\ 0, & i \neq true\,label \end{cases} \end{aligned} \tag{1}$$

where K is the number of classes, and p_i is the result of applying the softmax function to the vector output by the model.

This loss function only calculates the correct class, so it tends to ignore the relationship between the true label and other labels, resulting in a more arbitrary model. Therefore, we use the LabelSmooth strategy to adjust the loss function. It is a regularization method that

adds random noise to each dimension of the original one-hot representation. We add a label smoothing term to the loss function:

$$L_{smooth} = -1/n \sum_{i=1}^{K} log(p_i) \tag{2}$$

The final loss function is as follows:

$$L = (1 - \varepsilon)L_{normal} + \varepsilon L_{smooth} \tag{3}$$

where ε is the smoothing coefficient.

This method can properly pay attention to the inter-class relationship and avoid the problem of model overfitting. In this way, the model can have stronger generalization ability by suppressing the output difference between positive and negative samples.

4. Background

In this section, we first describe common definitions used in the literature of the website fingerprinting attacks. Then, we give a formal statement of the website fingerprint attack problem. This allows us to clearly recognize the constraints and form the evaluation criteria.

4.1. Definitions

In this section, we introduce the general threat model and assumptions of website fingerprinting attacks on Tor. Roughly, the attacker is assumed as a local passive eavesdropper. Here, the word 'local' means that the attacker can only listen to the communication between the client endpoint and the entry of relay nodes, as shown in Figure 1. Similarly, 'passive' represents that the attacker can only monitor the network packets but cannot insert, modify, or drop packets. Furthermore, potential attackers in the real world include but are not limited to the internet service providers (ISP), routers, and autonomous systems (AS) between the client and the entry of relay nodes, and local system administrators.

When executing the WF attack, since it is not convenient to identify all the websites that the user may visit, the attacker first selects a group of websites that they are interested in. The set of these websites is called the monitored websites. Likewise, the other websites are called the unmonitored websites. In the WF attack, the attacker first uses Tor to visit the monitored websites for multiple times and captures and traces all the packets. Then, the attacker extracts important features from the traced packets as the website fingerprints and uses these features to train a classifier. Finally, the attacker can collect new traffic and use the trained classifier to identify the target websites.

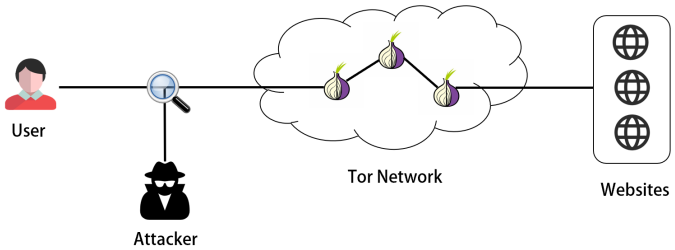


Figure 1. The threat model of the website fingerprinting attacks.

Closed-World Setting vs Open-World Setting: The WF attacks are usually evaluated in both the closed-world and open-world settings. The closed-world setting assumes that a user can access a limited number of websites, so that the attacker only needs to classify a specific set of websites. It is clear that users do not just visit a specific set of websites; their behavior in the real world is unpredictable. Consequently, the closed-world setting is not very practical [8]. However, since the closed-world setting is also a classic scenario for the

task of classification, the evaluation results under the closed-world setting are still used as indicators to evaluate the effect of the attack model.

Subsequent works also consider the open-world setting as a more realistic scenario. Since there are so many websites that users can visit, the attacker finds it hard to collect training data for all of the websites. Therefore, the open-world evaluation model allows an attacker to repeatedly collect traces of a small number of websites as a monitored set. Meanwhile, the attacker can also collect traces of other websites to form an unmonitored set. Samples from the unmonitored set are included in the training data as an additional label to help the classifier distinguish between monitored and unmonitored websites. In this paper, we will evaluate the attack effect in both scenarios, respectively.

Single Website vs Multiple Websites: Most of previous works assume that the Tor users visit only one website each time. This assumption is too ideal, and it obviously cannot represent the real visit behavior of users. For instance, a user may browse two different websites simultaneously. So, the visiting of multiple websites at the same time inevitably brings in overlapping times [25,26]. Some previous works [25,26] have explored the issue of multiple websites browsing, and we will not discuss it further in this paper.

Traffic Parsing: In the WF attacks, it is usually assumed that the attacker has the ability to distinguish the traffic data generated by visiting websites from other traffic data. If the attacker eavesdrops on the communication between the client and the Tor entry of relay nodes, then all the traffic via Tor is multiplexed across the requirement of the TLS protocol. However, previous works [27] implemented the parsing for multiplexed TLS traffic. So, we can simply follow the traffic parsing as before.

4.2. Problem Statement

The website fingerprinting attack problem aims to identify the website through the traffic trace generated by the behavior of visiting the website. Visiting a website y once will generate a traffic trace $\mathbf{x}$, and the deep learning model needs to use $\mathbf{x}$ as input to output a predicted website. The model is often trained through a labeled training set $D_{train} = (\mathbf{x}_i, y_i)_{i=1}^{M}$, where M is the number of training traces. M is usually large; however, in order to adapt to the deployment and performance requirements of real environments, some methods try to maintain effectiveness while reducing the number of training data. At the same time, since the test data distribution in the real environment is often different from the training data distribution, and new websites may be added, it is usually required that the model can quickly adapt to new attack tasks with only a few labeled training samples. Specifically, given a new training data set $D_{newtrain} = (\mathbf{x}_i, y_i)_{i=1}^{KN}$, which contains K new websites, and each website has N labeled traffic traces. This dataset is used to update the model to adapt it to new tasks.

In WF, a traffic trace of a website is usually represented as a two-tuple sequence <timestamp, ±packet_size>, where the sign before packet_size indicates the direction of the packet. The positive sign represents outgoing packets, and the negative sign represents incoming packets.

5. Problem Analysis

Prior works typically only use the packet direction but ignore packet size and timestamp. However, some recent works point out that timestamps can also reveal valid information. Intuitively, the direction and timing of packets are features of two different dimensions. Therefore, by using them together to train the model one should be able to achieve better results. We consider both the packet direction and the inter-packet time. Following the method of Wang et al. [2], we use +1 for outgoing packets and -1 for incoming packets, so the packet direction sequence can be represented as $D = (d_1, d_2, \ldots, d_L)$, where $d_i \in \{-1, +1\}$. For timing information, we do not directly use the timestamp of the packet but subtract the timestamp of the previous packet from the timestamp of the current packet to obtain a sequence of time intervals between packets, which can be represented by $T = (t_1, t_2, \ldots, t_L)$, where $t_i > 0$. Since the model requires a fixed-length input, we

determine a length threshold L, padding with zeros if the sequence length is less than L and truncating the first L elements if the sequence length is greater than L.

In TikTok [24], they multiplied the direction sequence with the timing sequence as input into the model. From their experimental results, it can be seen that the accuracy of the model has indeed improved, but the improvement is very small. Inputting the direction sequence or timing sequence alone can achieve good accuracy, but combining them directly produces only a small improvement, which shows that there is not a simple linear relationship between direction and timing. Therefore, directly merging them into the model cannot improve the classification accuracy to the greatest extent. Var-CNN [20] inputs the direction sequence and timing sequence into two models, respectively, and finally averages the output of the softmax layer of the two models to obtain the final result. However, according to common sense, the direction in which data packets are sent and the time interval between them will not be completely irrelevant; there should be some relationship between them, and this relationship can help the model to classify. This relationship is imperceptible, so we can first perform feature extraction on the direction sequence and timing sequence and then fuse their features to maximize the use of these two kinds of information.

In this paper, we propose a framework, including the basic model and the subsequent update of the model. At the same time, for the problem in which the distribution of training data and test data is different, we give the solution under different numbers of training data.

6. Website Fingerprinting Attack via CNN

6.1. System Overview

We provide an overview of the framework of the proposed schemes. As shown in Figure 2, the attack is mainly divided into three phases, including the training phase, attack phase, and fine-tuning phase.

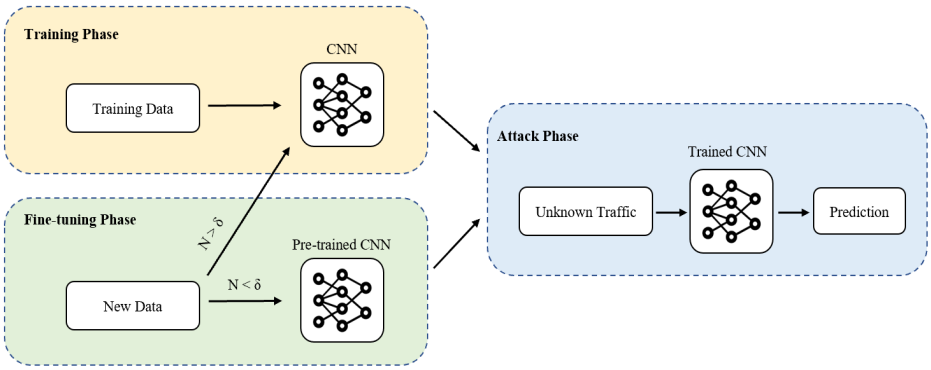


Figure 2. The schematic illustration for the framework of the WF attack.

Training Phase: The attacker creates a CNN model for WF attack. To train the attack model, the attacker needs to collect traffic data as a training set. Then, the training set is used to train the CNN model, which is later used as a classifier to perform classification tasks.

Attack Phase: The attacker uses the trained classifier to perform a WF attack. First, the attacker captures unknown traffic between the user and the entry node. Afterwards, the unknown traffic is fed into the trained classifier for classification to infer the targeted website of the traffic.

Fine-Tuning Phase: Since traffic patterns change from time to time, the trained model requires regular updates. The attacker needs to re-collect N examples for each monitored website. Considering the difficulty of traffic collection, N is usually set small (e.g., five examples per website). This is regarded as a threshold δ in the model. If N is less than δ, the attacker only needs to use new traffic data to fine-tune the parameters of the model. Otherwise, if the attacker collects more than δ examples for each website, the attacker can

choose to retrain the model. After the model is updated, it can continue to be used in the attack phase.

6.2. The CNN-Based WFA Scheme

The architecture of the CNN-based website fingerprinting attack (CWFA) scheme is shown in Figure 3. The CNN has a total of 12 convolutional layers, and each convolutional layer is followed by a BN layer and an activation layer. Similar to DF, we also follow the idea of large image classification and use two convolutional layers before the pooling layer to increase the network depth to ensure that the CNN model sufficiently learns the patterns.

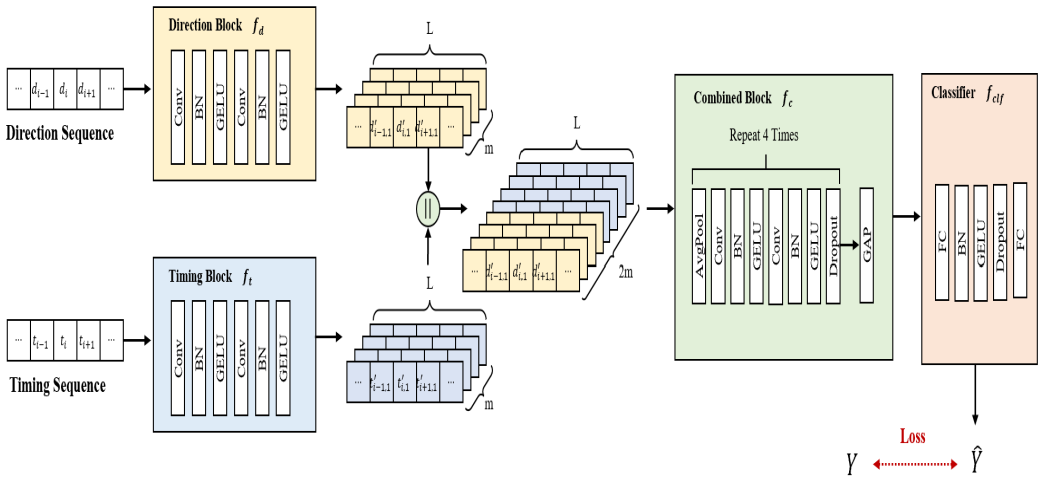


Figure 3. The architecture of the CNN-based website fingerprinting attack (CWFA) scheme.

In order to describe the architecture of the CWFA scheme more clearly, we divide the scheme into four blocks and use f_d, f_t , f_c, and f_{clf} to represent the four blocks respectively. The input $X = (D, T)$ contains direction and time sequences. At the beginning, the sequences D and T are input into the direction block and the timing block, respectively. The corresponding feature maps are obtained as below.

$$D^{'} = f_d(D), D^{'} \in \mathbb{R}^{m \times L} \tag{4}$$

$$T^{'} = f_t(T), T^{'} \in \mathbb{R}^{m \times L} \tag{5}$$

Here, m represents the number of feature maps. Next, $D^{'}$ and $T^{'}$ are concatenated and fed into the combined block.

$$\hat{Y} = f_{clf}(f_c(D^{'} \| T^{'})) \tag{6}$$

where $\|$ means concatenation and $\hat{Y}$ represents the probability that X belongs to a specific class. Compared with other blocks, the combined block adds a pooling layer before every two convolution blocks and adds a dropout layer after it. For a classfier, we use two FC layers, and the first FC layer is followed by a BN layer, an activation layer and a dropout layer. Before the classfier, the output of the convolution is converted to a vector by a global average pooling (GAP) layer, which can better integrate global spatial information and reduce the number of parameters.

Avoid Overfitting: CNN models are prone to overfitting. In order to prevent the occurrence of overfitting, we mainly take the following measures. First, we add a dropout layer at the end of each block in the combined feature extractor, and a BN layer is used after each convolutional layer. Dropout randomly hides units so that the model does not rely too much on certain local features. Second, the LabelSmooth strategy that we used in the loss function can also prevent overfitting.

Learning Rate Decay: The learning rate is an important parameter in the training of neural networks. The previous WF usually uses a fixed learning rate, but the fixed learning rate makes it difficult for the model to converge to the global optimal point. Therefore, we use the cosine annealing learning rate decay strategy. In the early stage of model training, a larger learning rate is used to optimize the model. As the number of iterations increases, the learning rate will gradually decrease to ensure that the model parameters will not fluctuate too much in the later stage of training, thereby speeding up the convergence of the model.

6.3. The Fine-Tuning-Based CWFA Scheme

Due to constant change of the traffic patterns, we established that the CWFA scheme cannot maintain the high accuracy all the time. Since it is very difficult to periodically re-collect large numbers of data, knowing how to use a small number of traffic data to make the model valid over a long period of time is important. In this work, we choose to design a fine-tuning mechanism to enable the CWFA scheme to support new data distributions.

The fine-tuning mechanism for the CWFA (FM-CWFA) scheme is mainly divided into three steps.

Step 1: The attacker needs to use a source dataset to train a robust model by the CWFA scheme. The trained CNN model in the CWFA scheme is viewed as a pre-trained model.

Step 2: When the traffic pattern changes, that is, the model cannot accurately identify websites, the attacker needs to collect a few samples for each website as the target dataset to adjust the parameters of the pre-trained model. Specifically, the direction block f_d, the timing block f_t, and the combined bloack f_c are fine-tuned with a small learning rate, and the classifer f_{clf} is initialized and trained with a large learning rate.

Step 3: The attacker can use the adjusted model to classify new unknown traffic.

7. Evaluation

In this section, we evaluate the performance of our model and compare it with state-of-the-art deep learning-based WF attacks.

7.1. Setting

To gain a more comprehensive understanding of the performance, we compare the CWFA and FM-CWFA schemes with five existing deep learning-based WFA schemes, namely, the DF, Var-CNN, TF, AF, and TLFA schemes. Here, we use the same datasets in the comparative experiments for fairness.

The hyper-parameters in the FM-CWFA schemes need to be tuned to stably maintain high accuracy. In order to select suitable hyper-parameters, the hyper-parameter space is first determined empirically. Then, the optimal value of each hyper-parameter is explored from the whole space to form the optimal hyper-parameter combination. The search space of hyper-parameters and their final selected values are shown in Table 1. In the experiment, we follow the previous common settings and take L as 5000.

7.2. Dataset

In the experiments, we use two kinds of datasets that are frequently used in the related works. The details of these datasets are listed as follows:

The Undefended DF Dataset [7]: This dataset was collected in 2016. Both monitored and unmonitored websites were selected from the Alexa top websites. For the closed-world setting, it contains 95 monitored websites with 1000 traces per website. For the open-world setting, it contains 40,000 unmonitored websites with only one trace per website.

The Defended DF Dataset [7]: From the viewpoint of defense strategies, we use two defended DF datasets, including the WTF-PAD dataset [7] and FRONT dataset [13].

The Undefended Wang Dataset [2]: This dataset was collected in 2013. The monitored websites were collected from websites blocked in China, and the unmonitored websites were collected from the Alexa top sites. For the closed-world setting, it contains

100 monitored websites with 90 traces per website. For the open-world setting, it contains 9000 unmonitored websites with only one trace per website.

The Defended Wang Dataset [2]: From the viewpoint of defense strategies, we use two defended Wang datasets, including the WTF-PAD dataset [7] and FRONT dataset [13].

When conducting experiments, the dataset is split into training, validation, and test sets. For each class, 10% of the monitored websites are used as the test set, 5% of the remaining data are used as the validation set, and the rest of the data are used as the training set. In the open-world experiment, the number of training and testing data varies depending on the concrete settings.

Table 1. Hyperparameter selection for model.

Hyperparameters	Search Range	Final
Optimizer	[Adam, Adamax, SGD]	SGD
Learning Rate	[0.001...0.01]	0.005
Training Epochs	[20...50]	30
Batch Size	[16...256]	[32...128]
Activation Functions	[Tanh, ReLU, GELU]	[GELU]
Number of Filters		
Block1_d[Conv1_d, Conv2_d]	[16...64]	[32, 32]
Block1_t[Conv1_t, Conv2_t]	[16...64]	[32, 32]
Block2[Conv3, Conv4]	[64...128]	[64, 64]
Block3[Conv5, Conv6]	[64...256]	[128, 128]
Block4[Conv7, Conv8]	[128...512]	[256, 256]
Block5[Conv9, Conv10]	[256...1024]	[512, 512]
Pooling Layers	[Average, Max]	Average
Dropout [Block, FC]	[0.1...0.7]	[0.1, 0.5]
Fine-Tuning Learning Rate	[0.0001...0.1]	[0.0005, 0.01]

7.3. Evaluation Metrics

In this experiment, we use accuracy, precision, and recall to evaluate the performance of the model. For the classification of a certain class, if a trace belongs to this class and it is correctly classified then it is called true positive (TP), and if it is misclassified then it is called false negative (FN). If a trace does not belong to this class and it is classified correctly, it is called true negative (TN), and if it is misclassified, it is called false positive (FP). The definitions of these three evaluation metrics are as follows:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

7.4. The Evaluation of the CWFA Scheme

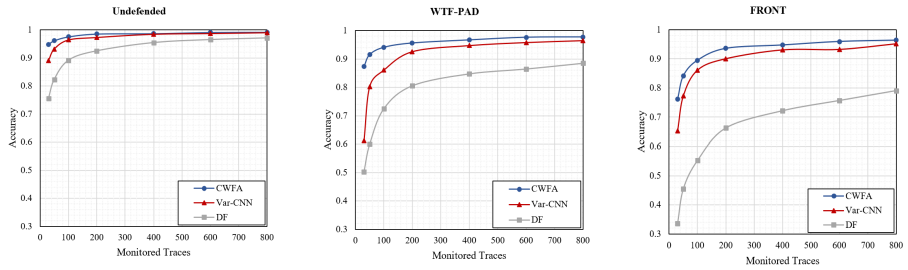
First, we evaluate the performance of the WF attack in the scenario where the model is trained from scratch using the dataset. We evaluate the undefended and defended DF dataset and use DF and Var-CNN to compare with our schemes.

7.4.1. Closed-World Setting

We evaluate the performance of the model in the closed-world setting on the undefended dataset and defended datasets, respectively. Although this setting is not realistic, it is suitable for comparing the performance of different classifiers. In the closed-world setting, WF is regarded as a multi-classification task. We use accuracy, average precision, and average recall as the performance metrics to evaluate the classification effect of the

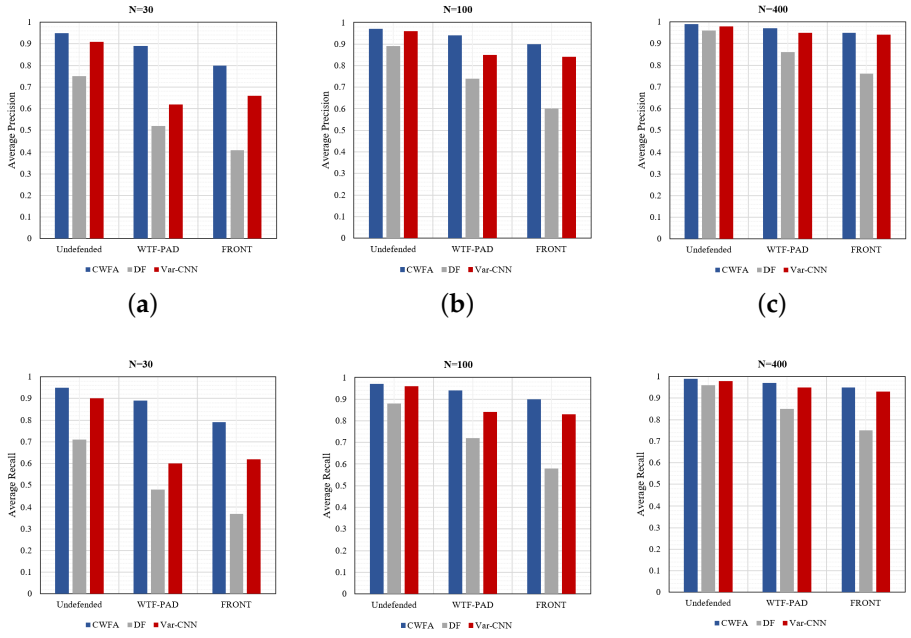
model in the experiment. At the same time, the experiment also considers the time cost of model training.

In the experiment, the impact of different numbers of training traces on the model performance was considered. Figure 4 shows the variation in the accuracy of the three models on the undefended and defended datasets as the training traces of the monitored website increase, and the training time for three models with increasing training data, where each model is trained for 30 epochs. Figure 5 shows the average values of precision and recall for all classes with N training traces per site.



(**a**) (**b**) (**c**)

Figure 4. CWFA evaluation: closed-world accuracy. (**a**) Undefended dataset (**b**) WTF-PAD dataset (**c**) FRONT dataset.



(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

Figure 5. CWFA evaluation: the average precision and recall in the closed-world setting. (**a**) Average precision when N = 30 (**b**) Average precision when N = 100 (**c**) Average precision when N = 400 (**d**) Average recall when N = 30 (**e**) Average recall when N = 100 (**f**) Average recall when N = 400.

It can be seen from the results that our model is able to achieve better classification performance than DF and Var-CNN, and this performance advantage becomes more pronounced as the number of training data decreases . For the undefended dataset, the accuracy of the CWFA scheme can reach 95% when there are only 30 traces per website and 99% when there are 800 traces per website. For defended datasets, the results show that the CWFA schme is still effective, and the advantage is more obvious than that in the undefended case. For example, with 100 traces per monitored website, the accuracy of the CWFA scheme can achieve 94%, while the accuracy of the Var-CNN and DF schemes is

approximately 86% and 72%, respectively. Var-CNN and DF require more data to achieve the same results as our attack. This difference becomes larger as the training data decreases.

The ability of DF to resist defense strategies is significantly worse, which means that for the traces protected by the defense, the effective information of direction information leakage is hidden to a certain extent. Adding timing information can effectively improve the performance of the learning model. Especially for FRONT, the timing information leaks more effective information about the website, allowing the attack to successfully destroy the defense.

Figure 6 shows the training time for three models with increasing training data, where each model is trained for 30 epochs. It can be seen that the training time of Var-CNN is significantly higher than the other two methods. This is because Var-CNN trains two dilated causal ResNets, and the structure of the network is relatively more complex, resulting in a large time overhead. The training time of our model is much lower than Var-CNN and slightly higher than DF. Considering the accuracy and time overhead, we can conclude that our method outperforms DF and Var-CNN.

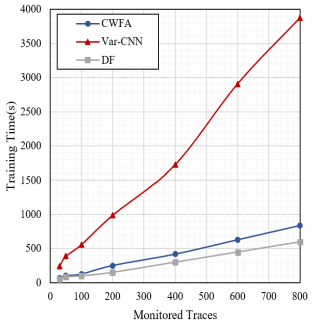


Figure 6. CWFA evaluation: running time of different schemes in the training phase.

7.4.2. Open-World Setting

We now evaluate the performance of the attack in the open-world setting. Unlike the closed-world setting, the attacker needs to determine whether the traffic trace is coming from a monitored website or an unmonitored website here. Therefore, we focus on binary classification, which means determining whether a traffic trace is being monitored. According to the discussion of open-world evaluation metrics in the previous WF literature [3,28], using the true-positive rate (TPR) and false-positive rate (FPR) as evaluation metrics may lead to an inappropriate interpretation of attack performance due to imbalanced data in the real world. Therefore, we use precision and recall to evaluate the performance of the model to circumvent the benchmark rate fallacy.

The training set in this experiment contains 28,500 monitored traces (300 traces per website for 95 monitored websites) and 30,000 unmonitored traces (one trace per website), and the test set contains 4750 monitored traces (50 traces per website for 95 monitored websites) and 10,000 unmonitored traces (one trace per website), where the unmonitored traces for the training and test sets do not overlap.

Figure 7 shows the precision-recall curves of the three WF attacks. On the undefended dataset, when tuned for high precision, the precision and recall of the CWFA scheme can reach 0.99 and 0.8, respectively. When tuned for high recall, its precision and recall can achieve 0.82 and 0.99, respectively. In the face of traces that have been defended, these attacks are all less effective, but the CWFA scheme still has advantage. On the WTF-PAD dataset, when tuned for high precision, the precision and recall of the CWFA scheme can reach 0.97 and 0.6, respectively. When tuned for high recall, its precision and recall can achieve 0.66 and 0.98, respectively. On the FRONT dataset, when tuned for high precision, the precision and recall of the CWFA scheme can reach 0.94 and 0.5, respectively. When tuned for high recall, its precision and recall can achieve 0.5 and 0.98, respectively.

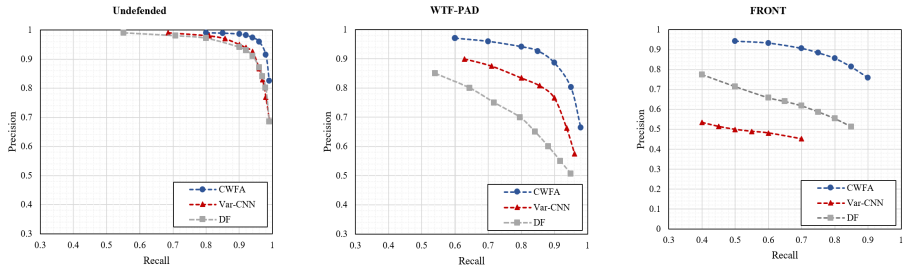


Figure 7. CWFA evaluation: open-world precision-recall curves. (**a**) Undefended dataset (**b**) WTF-PAD dataset (**c**) FRONT dataset.

7.5. The Evaluation of the FM-CWFA Scheme

Next, we evaluate the performance of the WF attack in the scenario where the pre-training and classification datasets are collected at different times, and there are only a few traces of each website in the classification dataset.

7.5.1. Closed-World Setting

In this experiment, we consider the case where very little training data are available for each class to update the model. From Table 2, it can be seen that when the training data are small enough, the accuracy of the classification will drop rapidly. When there are only a few examples per class, it is almost meaningless to directly train the model.

Table 2. Accuracy of training with a small number of data.

Dataset	Number of Traces				
	1	**5**	**10**	**15**	**20**
Undefended	12.0	71.8	86.6	91.2	93.3
WTF-PAD	6.1	44.5	71.7	80.0	83.8
FRONT	5.0	22.7	41.5	56.0	66.7

Next, we evaluate whether the performance of the CWFA scheme can be improved by the fine-tuning mechanism. We use the Wang dataset for pre-training and the DF dataset for training and testing. The two datasets were collected three years apart, and they use different versions of the Tor browser. During the pre-training phase, we sample 25 examples for each website in the Wang dataset for the pre-training the model. During the training phase, we sample N = 1, 5, 10, 15, and 20 examples for each website in the DF dataset to fine-tune the model. During the testing phase, we use 100 examples per website to test the performance of the FM-CWFA scheme.

In this experiment, we use three previous methods to compare with the FM-CWFA scheme, namely, TF, AF, and TLFA. The feature extractor used by TLFA is similar to DF, and the above experiments have shown that the CWFA scheme proposed in this paper is better than DF. Therefore, in order to more fairly compare the effectiveness of the fine-tuning mechanism used by TLFA and FM-CWFA in the field of WF attacks, we replace the feature extractor in the original TLFA method with the CWFA scheme proposed in this paper. We denote this improved variant as TLFA*. At the same time, for a fairer comparison, all of the methods in the experiment use the same number of data in the pre-training phase.

Table 3 shows the performance of the FM-CWFA scheme. We conduct experiments on undefended, WTF-PAD, and FRONT datasets, respectively. The results show that the FM-CWFA scheme is effective for improving the performance no matter whether defense strategies are adopted. For example, compared to the results in Table 2, when there is only five example per website, the accuracy can be increased by 10% without defense, 26%

with WTF-PAD, and 22% with FRONT. It can also be seen that FM-CWFA outperforms TF, AF, and TLFA no matter what kind of defense is used or what number of data are used, which illustrates the superiority of the fine-tuning mechanism proposed in this paper. Fine-tuning has an upper limit on the improvement of the performance. When there are more than 20 examples per website, the performance of the FM-CWFA scheme no longer improves. However, in this case, the attacker can directly retrain the attack model instead of fine-tuning it to achieve good results.

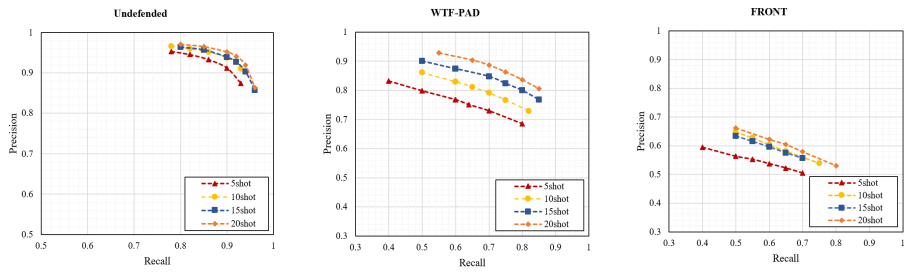
Table 3. FM-CWFA evaluation: closed-world results.

Dataset	Methods	Number of Traces				
		1	**5**	**10**	**15**	**20**
Undefended	TF	40.2	59.8	64.7	67.4	67.5
	AF	24.3	63.1	74.5	84.7	85.4
	TLFA*	51.8	76.8	83.9	86.7	88.1
	FM-CWFA	52.2	81.8	88.6	92.2	93.3
WTF-PAD	TF	15.1	31.5	34.3	36.3	38.0
	AF	7.9	27.4	38.1	52.4	55.7
	TLFA*	31.9	60.1	68.2	72.4	74.0
	FM-CWFA	36.5	70.3	79.2	83.5	85.4
FRONT	TF	8.7	13.4	15.8	16.7	17.3
	AF	4.1	15.0	22.3	33.8	35.4
	TLFA*	16.7	36.5	45.7	50.0	52.7
	FM-CWFA	17.5	45.0	62.0	69.6	73.8

7.5.2. Open-World Setting

In the following experiments, we explored the effect of the fine-tuning mechanism in the open-world setting. We still use the Wang dataset for pre-training, and the DF dataset for training and testing. We use the monitored websites in the Wang dataset to pre-train the model, mainly because of the wide variety of monitored websites. The important thing in the pre-training phase is to learn how to extract the features of the website, so more classes of training data win more benefits.

Figure 8 shows the precision-recall curves of the FM-CWFA scheme with and without defenses. The open-world results in undefended and WTF-PAD defended are relatively effective. However, with the FRONT defense, the result of the attack in the open-world is not as effective as others, which means that the defense effect of FRONT in this experiment is relatively good. It can also be seen from the results that as the labeled data of the target dataset increases, the attack results are better.



(**a**) (**b**) (**c**)

Figure 8. FM-CWFA evaluation: open-world precision-recall curves. (**a**) Undefended dataset (**b**) WTF-PAD dataset (**c**) FRONT dataset.

7.6. Technical Evidence and Explanation of the Success

7.6.1. Analysis of Features and Model Architecture

In this experiment, in order to evaluate the impact of features and CNN model architecture on classification accuracy, we used the following settings.

Setting1: The purpose of this setting is to analyze the importance of using both direction and time features. In order to explore the impact of these two features on the attack effect, we input the direction sequence and timing sequence to model separately and call these two cases Setting1-direction and Setting1-timing, respectively. Since the model has only one input and the direction block and the timing block have the same structure, we ignore one of the blocks for training.

Setting2: The purpose of this setting is to analyze whether the direction information and timing information have a simple linear relationship. In this setting, we multiply the direction sequence with the timing sequence before feeding into the model. Again, since there is only one input, we ignore the direction block or the timing block.

Setting3: The purpose of this setting is to analyze whether direction and timing features are independent. This setting treats direction and timing as two independent features, and the direction sequence and timing sequence are fed into the two identical models separately. Finally, the probabilities of each category output by the two models are added to obtain the final classification result.

Setting4: The purpose of this setting is to analyze how features are fused best for shallow features in direction and timing. This setting is the CWFA scheme proposed in this paper. We discuss several methods of feature fusion, including addition, multiplication, and concatenation, called Setting4-add, Setting4-multiply, and Setting4-concat, respectively.

Results: Table 4 shows the results under different settings. From the results of Setting1, it can be seen that the result of using only one feature is significantly worse than using two features. Interestingly, previous work tends to think that direction is the more effective feature, but the experimental results show that when the trace is defended, the model trained by timing sequence alone can have the same or even better performance than the model trained by direction sequence, especially on the FRONT dataset. Therefore, the timing sequence tends to leak more information for traces defended by FRONT.

Table 4. Model accuracy under different settings.

Setting	Dataset		
	Undefended	**WTF-PAD**	**FRONT**
Setting1-direction	97.3	89.0	80.2
Setting1-timing	95.9	89.3	89.8
Setting2	96.9	90.2	87.3
Setting3	98.2	93.8	91.9
Setting4-add	97.7	95.4	93.4
Setting4-multiply	98.3	95.4	92.2
Setting4-concat	98.5	95.6	93.6

Using the same two features, the results of Setting4 are generally better than Setting2 and Setting3. This shows that the direction and timing sequence are not completely independent; there is a certain relationship between them, but this relationship is not a simple linear relationship. Therefore, the best effect can be achieved by learning and extracting this relationship through CNN and then fusing the features. It can be seen from the results that different feature fusion methods have little impact on the attack effect, and the concat method used in the CWFA scheme is slightly higher than other methods.

7.6.2. Analysis of Features Sequence Information

Since the neural networks are able to classify websites by sequences of packet direction and time interval, we wonder what information the neural network relies on in these

sequences to classify. According to common sense, the timing and order in which both parties send packets are important in network communication. So we try to shuffle the direction sequence and time sequence, and then we input them into the neural networks to observe the classification results. The results are shown in Table 5.

Table 5. Sequence random test results (accuracy).

Dataset	Methods	Normal	Random
Undefended	DF	0.89	0.30
	Var-CNN	0.96	0.48
	CWFA	0.97	0.52
WTF-PAD	DF	0.73	0.14
	Var-CNN	0.86	0.25
	CWFA	0.94	0.29
FRONT	DF	0.55	0.06
	Var-CNN	0.86	0.17
	CWFA	0.89	0.15

The results show that once the two sequences are shuffled, the accuracy of the attack drops significantly, which means that the classification of the website depends on the order of packets and time intervals between packets. Even if the content of the website changes or the browser version changes, the sequence order is still effective in distinguishing different websites, so the AI model trained on the old dataset will help the classification of the new dataset.

8. Discussion

In this work, we are able to conduct successful attacks using fewer data and mitigate the concept drift problem. Meanwhile, several defense strategies, such as WTF-PAD, are ineffective against our attacks, which means that if defenders want to prevent WF attacks, more powerful defense methods are required.

Our attack demonstrates that defenses against WF attacks by adding dummy packets are vulnerable. These methods do not perform any delay on real packets. The timestamps of real packets are still exposed, and the classifier can find valuable information from these timestamps. Therefore, future WF defense strategies can try to cut down the valid information hidden in the timestamps as much as possible by adding a reasonable packet with appropriate time delay.

For future attacks, there are some issues that need to be paid attention to. Firstly, existing WF attacks usually achieve a high recall rate and a low false-positive rate under the condition of high base rate pages, but the proportion of the number of unmonitored websites and the number of monitored websites in the experiment cannot fully simulate reality. Therefore, the classifier is prone to generate a large number of false positives, resulting in unreliable classification results. Secondly, there has been a lot of research focus on the issue of the multi-tab problem, but the effect cannot meet the standards of actual use, so more effective methods still need to be explored. Knowing how to break through the limitations of current researches to improve the possibility of practical application is still an issue that should be considered in future research.

9. Conclusions

In this paper, we propose a new CNN-based website fingerprinting attack scheme that can effectively automatically extract features from packet directions and time intervals. Then, a fine-tuning scheme is proposed that can support changes in the statistical patterns. The proposed scheme has better performance and stability and can handle fewer training data in both closed-world and open-world environments. Furthermore, we analyze the

technical reason for the success and designed experiments to prove the inevitability of the success. Finally, we give future directions for website fingerprinting attacks and defenses.

Author Contributions: Conceptualization, T.P.; methodology, T.P. and Z.T.; investigation, T.P. and Z.T.; validation, T.P. and D.X.; writing—original draft preparation, T.P.; and writing—review and editing, Z.T. and D.X. All of the authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Natural Science Foundation of China (NSFC) under grant No. 62172040 and the National Key Research and Development Program of China under grant No.2021YFB2701200.

Data Availability Statement: Data will be made available on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dingledine, R.; Mathewson, N.; Syverson, P.F. Tor: The Second-Generation Onion Router. In Proceedings of the USENIX Security Symposium, San Diego, CA, USA, 9–13 August 2004; pp. 303–320.
2. Wang, T.; Cai, X.; Nithyanand, R.; Johnson, R.; Goldberg, I. Effective attacks and provable defenses for website fingerprinting. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, 20–22 August 2014; pp. 143–157.
3. Panchenko, A.; Lanze, F.; Pennekamp, J.; Engel, T.; Zinnen, A.; Henze, M.; Wehrle, K. Website Fingerprinting at Internet Scale. In Proceedings of the NDSS, San Diego, CA, USA, 21–24 February 2016.
4. Hayes, J.; Danezis, G. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 1187–1203.
5. Abe, K.; Goto, S. Fingerprinting attack on Tor anonymity using deep learning. *Proc. Asia-Pac. Adv. Netw.* **2016**, *42*, 15–20.
6. Rimmer, V.; Preuveneers, D.; Juarez, M.; van Goethem, T.; Joosen, W. Automated Website Fingerprinting through Deep Learning. In Proceedings of the 25th Annual Network and Distributed System Security Symposium, NDSS, San Diego, CA, USA, 18–21 February 2018; pp. 18–21.
7. Sirinam, P.; Imani, M.; Juarez, M.; Wright, M. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1928–1943.
8. Juarez, M.; Afroz, S.; Acar, G.; Diaz, C.; Greenstadt, R. A critical evaluation of website fingerprinting attacks. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 263–274.
9. Sirinam, P.; Mathews, N.; Rahman, M.S.; Wright, M. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 1131–1148.
10. Wang, C.; Dani, J.; Li, X.; Jia, X.; Wang, B. Adaptive fingerprinting: Website fingerprinting over few encrypted traffic. In Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy, Virtual Event USA, 26–28 April 2021; pp. 149–160.
11. Chen, M.; Wang, Y.; Qin, Z.; Zhu, X. Few-shot website fingerprinting attack with data augmentation. *Secur. Commun. Netw.* **2021** *2021*, 2840289. [CrossRef]
12. Juarez, M.; Imani, M.; Perry, M.; Diaz, C.; Wright, M. Toward an efficient website fingerprinting defense. In Proceedings of the European Symposium on Research in Computer Security, Heraklion, Greece, 26–30 September 2016; pp. 27–46.
13. Gong, J.; Wang, T. Zero-delay lightweight defenses against website fingerprinting. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 717–734.
14. Dyer, K.P.; Coull, S.E.; Ristenpart, T.; Shrimpton, T. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 332–346.
15. Cai, X.; Nithyanand, R.; Wang, T.; Johnson, R.; Goldberg, I. A systematic approach to developing and evaluating website fingerprinting defenses. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 227–238.
16. Cai, X.; Nithyanand, R.; Johnson, R. Cs-buflo: A congestion sensitive website fingerprinting defense. In Proceedings of the 13th Workshop on Privacy in the Electronic Society, Scottsdale, AZ, USA, 3 November 2014; pp. 121–130.
17. Wang, T.; Goldberg, I. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver BC Canada, 16–18 August 2017; pp. 1375–1390.
18. Herrmann, D.; Wendolsky, R.; Federrath, H. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In Proceedings of the 2009 ACM Workshop on Cloud Computing Security, Chicago, IL, USA, 13 November 2009; pp. 31–42.

19. Panchenko, A.; Niessen, L.; Zinnen, A.; Engel, T. Website fingerprinting in onion routing based anonymization networks. In Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, Chicago, IL, USA, 17 October 2011; pp. 103–114.
20. Bhat, S.; Lu, D.; Kwon, A.; Devadas, S. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. *Proc. Priv. Enhancing Technol.* **2019**, *2019*, 292–310. [CrossRef]
21. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
23. Chen, M.; Wang, Y.; Xu, H.; Zhu, X. Few-shot website fingerprinting attack. *Comput. Netw.* **2021**, *198*, 108298. [CrossRef]
24. Rahman, M.S.; Sirinam, P.; Mathews, N.; Gangadhara, K.G.; Wright, M.K. Tik-Tok: The Utility of Packet Timing in Website Fingerprinting Attacks. *Proc. Priv. Enhancing Technol.* **2020**, *2020*, 5–24. [CrossRef]
25. Yin, Q.; Liu, Z.; Li, Q.; Wang, T.; Wang, Q.; Shen, C.; Xu, Y. Automated Multi-Tab Website Fingerprinting Attack. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3656–3670. [CrossRef]
26. Cui, W.; Chen, T.; Fields, C.; Chen, J.; Sierra, A.; Chan-Tin, E. Revisiting assumptions for website fingerprinting attacks. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, Auckland, New Zealand, 9–12 July 2019; pp. 328–339.
27. Wang, T.; Goldberg, I. On Realistically Attacking Tor with Website Fingerprinting. *Proc. Priv. Enhancing Technol.* **2016**, *2016*, 21–36. [CrossRef]
28. Wang, T. High precision open-world website fingerprinting. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–20 May 2020; pp. 152–167.