Pattern Recognition 130 (2022) 108739

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Few-shot Website Fingerprinting attack with Meta-Bias Learning

Mantun Chen[a], Yongjun Wang[a,*], Xiatian Zhu[b]

[a] College of Computer, National University of Defense Technology, Changsha 410073, China
[b] Surrey Institute for People-Centred Artificial Intelligence, and the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Stag Hill, University Campus, Guildford GU2 7XH, UK

ARTICLE INFO

Article history:
Received 6 March 2021
Revised 11 February 2022
Accepted 23 April 2022
Available online 30 April 2022

Keywords:
User privacy
Internet anonymity
Data traffic
Website fingerprinting
Deep learning
Neural network
Few-shot learning
Meta-learning
Parameter factorization

ABSTRACT

Website fingerprinting (WF) attack aims to identify which website a user is visiting from the traffic data patterns. Whilst existing methods assume many training samples, we investigate a more realistic and scalable few-shot WF attack with only a few labeled training samples per website. To solve this problem, we introduce a novel *Meta-Bias Learning* (MBL) method for few-shot WF learning. Taking the meta-learning strategy, MBL simulates and optimizes the target tasks. Moreover, a new model parameter factorization idea is introduced for facilitating meta-training with superior task adaptation. Expensive experiments show that our MBL outperforms significantly existing hand-crafted feature and deep learning based alternatives in both closed-world and open-world attack scenarios, at the absence and presence of defense.

1. Introduction

For privacy protection in Internet use, anonymity networks have played an increasingly more important role. Among existing techniques, Tor network [1] is one of the best choices [2]. Security is always not absolute. It is found that the patterns of data transportation before reaching Tor servers can leak critical user information, although the web data themselves are encrypted. For instance, this can be done by a local passive attacker through capturing furtively the traffics between a user and the guard node of Tor networks, with the attacking positions including any devices in the same local area network (LAN) or wireless network, switch, router, and compromised Tor guard node. With such data, the attacker is likely to reason about which website a target user is visiting. As a type of side channel attack, this approach is often known as *website fingerprinting* (WF) attack [3].

Specifically, to implement a WF attack automatically, the attacker needs to first collect and label clean fingerprints for every individual website, and then design/learn informative features of these fingerprints for accomplishing attack via machine learning methods. Earlier methods rely on hand-crafting features manually designed with expert domain knowledge and have continuously improved the attacking performances [3–12]. A common weakness of these methods is that their features are highly sensitive to data content (*e.g.*, characteristics and implementation details of Tor), network environment, and Tor Browser Bundle. This restricts their usability in real applications. In contrast to manually-designed features, deep learning can totally avoid the slow and tedious process of accumulating domain knowledge by automatically learning feature representations from the training data [13]. Motivated by this favourable property, several latest state-of-the-art studies [14–16] have attempted to leverage deep learning for more scalable WF attack.

Existing deep learning methods can work well only when a sufficiently large number (*e.g.*, hundreds) of training samples are available for every target website. That is data hungry. Given small training data, deep models are not necessarily superior [14]. Importantly, this means not only a lengthy and expensive annotation process, but also an artificial assumption that all target websites are known in advance. In contrast, a real-world WF attack system needs to deal with *new* websites frequently due to the dynamic change in attacking requirements and conditions. This leads to a demand that the attacking method can be adapted to new tasks rapidly without a need to collect a big training set and re-train the model from scratch every time a new task arrives.

* Corresponding author.
E-mail addresses: chenmantun19@nudt.edu.cn (M. Chen), wangyongjun@nudt.edu.cn (Y. Wang), xiatian.zhu@surrey.ac.uk (X. Zhu).

https://doi.org/10.1016/j.patcog.2022.108739
0031-3203/

To meet such real-world requirements, a more realistic and scalable problem setup is *few-shot* WF attack where only a handful of labeled training samples per new class/website needs to be collected. This setup is indeed studied for the first time by a recent method Triplet Fingerprinting (TF) [17]. Specifically, TF trains a feature extraction model with triplet based ranking objective function and applies the nearest neighbour classifier in the trained feature space. This is inefficient in exploiting large-scale auxiliary training data and tackling the mismatch problem between the training and test settings. Later, Chen et al. [18] explored the data augmentation idea to boost the performance of deep learning WF attack methods. However, this method is unsatisfactory in small training data cases (*e.g.*, less than 10 training samples per class). To overcome this limitation, Transfer Learning Fingerprinting Attack (TLFA) [19] was recently proposed by training a strong embedding model and leveraging classical machine learning models (*e.g.*, SVM) on top. Nonetheless, this method remains inferior in the 1-shot case especially for the open-world setting.

To address the aforementioned limitations, we introduce a novel *Meta-Bias Learning* (MBL) method in a meta-learning principle. Instead of learning a deep metric model, we simulate and optimize the setting of target tasks explicitly via episodic training. Moreover, a new parameter factorization idea is introduced so that only a small fraction needs meta-training. This in turn allows to more effectively exploit large pre-training data via learning re-usable feature representation across different tasks. We adopt a more scalable classifier design as compared to the triplet metric learning loss. Altogether, our method can transfer more useful knowledge from large-scale auxiliary training data to different new tasks with limited labels.

We summarize the **contributions** of this paper:

(I) We investigate the under-studied, more realistic, and more challenging *few-shot website fingerprinting attack* problem. Beyond the existing metric learning [17], data augmenting [18] and transfer learning [19] attempts, we propose a more advanced meta-learning approach. Crucially, we further focus on the model learning scalability for knowledge transfer efficacy, and model optimization strategy for task adaptation capability.

(II) We propose a novel *Meta-Bias Learning* (MBL) method for solving few-shot WF attack. Specifically, we introduce a notion of parameter factorization, which avoids the need of meta-training the whole model. With this design, a majority fraction of parameters can be allocated to learn generic reusable feature representations useful for all different tasks, whilst the remaining used for more effective task adaptation.

(III) Extensive experiments show that our MBL outperforms significantly previous state-of-the-art methods in both closed-world and open-world few-shot WF attack scenarios, with and without defense. In particular, we also introduce new performance metrics for the open-world setting with more comprehensive measurements. Crucially, it is shown that MBL is the only method that can operate strongly for open-world 1-shot settings.

2. Related work

2.1. Problem objectives, scenarios and assumptions

The objective of WF attack is to identify which website a victim user is interacting with among a set of monitored target websites. By considering each website as a unique class, it is essentially a multi-class classification problem. There are several different scenarios with specific assumptions. The most common scenario is *closed-world* WF attack, where the victim user is assumed to only visit a set of *known* target websites under monitoring. This assumption however is not realistic, and therefore discarded in the *open-world* scenario. In this more realistic scenario, the victim user is considered to likely visit any websites, including those monitored ones, as typically experienced in most real-world applications.

Unlike the two above settings, a third scenario additionally considers *attacking defense* where the user takes some actions to defend against a potential attack. This would lead to higher attack difficulty. Representative defense techniques include Buflo [20], Tamaraw [21], Walkie-Talkie [22], WTF-PAD [23], FRONT [24], GLUE [24] and TrafficSliver [25]. Among them, WTF-PAD is considered arguably as one of the most promising defense methods for Tor networks due to low bandwidth overhead and zero delay, although it is still in the experimentation stage. Recently, an even stronger defense method called FRONT [24] was developed, characterized by a zero-delay property. We consider both WTF-PAD and FRONT defenses in our evaluations.

In the literature, several common assumptions are typically made. We briefly discuss three main aspects in the following. In terms of *user behavior*, it is assumed that all Tor users browse websites sequentially and only interact with a single website at a time. This simplifies the data traffic patterns. In terms of *background traffic data*, the attacker is assumed to be able to collect all the clean traces generated during the victim's visits against dynamic background data. This is increasingly possible as shown in [26], the multiplexed TLS traffic can be split into individual encrypted connections to each website. In terms of *network condition*, the attacker is assumed to have the same conditions as the victim user, including traffic condition and browser settings. To compare with the benchmark results, we follow these general assumptions for fair evaluations.

In this study, we focus on addressing the following challenge. Often, the attacker assumes that the training data have a similar distribution as the deployment data, *e.g.*, the training and testing websites are identical. This is a particularly strong and artificial assumption that often does not stand. This is because the network condition and task requirement are actually changing and evolving frequently. With the realistic dynamic property, the attacker needs to update the training data in order to have a robust attacking model over time. To that end, the attacker must collect a large set of training data at each time, which however, is infeasible and non-scalable due to high acquiring costs. Unfortunately, existing WF attack works [6,7,9,12,14–16] often ignore this aspect by assuming the availability of large training data all the time.

Considering that a large number of training samples from non-target websites can be easily collected in advance, we study the *few-shot learning* setting in WF attack. Specifically, we only assume only a handful of labeled training samples for every target (test) website. To enable model optimization, an independent large training set collected from other websites are often used for learning task-agnostic knowledge. This meets more real-world requirements, which however is largely under-studied in the WF attack literature.

2.2. Website fingerprinting attack methods

For Tor networks, the development of website fingerprinting attack methods can be divided into three categories: traditional website fingerprinting [6,7,9,11,12], deep website fingerprinting [14–16], and few-shot website fingerprinting [17–19], as summarized in Table 1. The first category utilizes largely inferior features manually designed based on the attacker's knowledge about anonymous traffic and applies traditional machine learning algorithms. Recently, with the fast advance of deep learning [13], the advan-

Table 1
Representative website fingerprinting attack methods. ML: Machine Learning; DL: Deep Learning; ETD: Extra Training Data.

Paper	Technique	Input type	Model type	ETD
Herrmann et al. 2009 [6]	Multinomial Naïve Bayes Classifier	The normalized frequency distribution of observable IP packet sizes	ML	No
Wang and Ian. 2013 [12]	Distance-based Metrics, Support vector machine	Tor cells	ML	No
k-NN, Wang et al. 2014 [10]	k-Nearest Neighbour classifier	A large feature set with weight adjustment	ML	No
CUMUL, Panchenko et al. [12]	Support Vector Machine (SVM)	The cumulative representation of a trace	ML	No
k-FP, Hayes and Danezis. 2016 [11]	k-Nearest Neighbour classifier, random decision forests	Packets statistics	ML	No
DF, Sirinam et al. 2018 [15]	VGG-like network	Directional sequence of Tor cells	DL	No
Var-CNN, Bhat et al. 2019 [16]	ResNet-18, Dilated causal convolutions	Ensemble of timing and direction, Cumulative statistical features	DL	No
HDA, Chen et al. 2021 [18]	Data augmentation	Directional sequence of Tor cells	DL	No
TF, Sirinam et al. 2019 [17]	Triplet network, Semi-hard triplet mining and k-Nearest Neighbour classifier	Directional sequence of Tor cells	ML and DL	Yes
TLFA, Chen et al. 2021 [19]	Transfer learning fingerprinting attack with different classifiers	Directional sequence of Tor cells	ML and DL	Yes
MBL (Ours)	Meta-Bias Learning with model parameter factorization	Directional sequence of Tor cells	DL	Yes

tage of feature representation learning starts to show great potentials in WF attack [14]. Different network architectures have been designed and explored to exceed the performance of traditional website fingerprinting methods. Despite the encouraging performance of deep website fingerprinting, it has several disadvantages, *e.g.*, data hungry or needing a large training set for every target website. Few-shot learning is therefore proposed to solve this limitation [17–19].

Traditional website fingerprinting attack methods. The first pioneer attack against Tor networks was evaluated by Herrmann [6] in 2009. It can only achieve an accuracy of 2.96% using around 20 training samples per website in the closed-world scenario. Later on, Wang and Ian [9] propose to represent the traffic data using more fundamental Tor cells (*i.e.*, direction data) as a unit rather than TCP/IP packets. This representation is meaningful and informative as it encodes essential characteristics of Tor data. By training an SVM classifier with a distance-based kernel, a groundbreaking performance with 90.9% accuracy was achieved on 100 sites each with 40 training samples. Recently, Panchenko et al. [12] propose an idea of sampling the features from a cumulative trace representation and achieve 91.38% accuracy with 90 training instances per website, which is named as CUMUL. Hayes and Danezis [11] exploit random decision forests to achieve similar results, known as k-fingerprinting (k-FP). A typical design of these above methods is a two-stage strategy including feature design and classifier learning. In general, this approach is not only constrained by the limitations of hand-crafted features but also lacks interaction between feature representation and classifier model, making the model performance inferior. **Deep website fingerprinting attack methods.** Motivated by the remarkable success of deep learning techniques in computer vision, natural language processing, and other fields [27–36], several deep learning WF attack methods have been introduced [14–16]. This paradigm can well solve the aforementioned weaknesses as discussed above. Instead of designing features based on domain knowledge, deep learning methods carry out feature representation learning and classification optimization from the raw training data *end-to-end*, without the need for mastering rich domain knowledge. For example, using VGG-like network [37] as the backbone, Sirinam et al. [15] propose a Deep Fingerprinting (DF) attack model and attain 98.3% accuracy on 95 websites. However, this method needs a large training set (*e.g.*, 1000 training samples per website), otherwise, it will suffer from significant performance drop. When using 50 training samples per website, DF can only hit around 90% accuracy. To overcome this limitation, Bhat et al. [16] developed the Var-CNN model using ResNet [38] and dilated causal convolution [39,40]. When small training sets (*e.g.*, 100 samples per website) are available, it achieves superior performance over DF. Nonetheless, it is also dependent on less stable time based features and less scalable hand-crafted statistical information. So it is not able to solve the limitations of earlier alternatives. Besides, It's worth mentioning that Wang et al. [41] made progress on the cross-platform website fingerprinting attack problem, which is orthogonal to the focus of our work. **Few-shot website fingerprinting attack methods.** There are three main few-shot website fingerprinting attack methods: Triplet Fingerprinting (TF) [17], Harmonious Data Augmentation (HDA) [18], and Transfer learning Fingerprinting Attack (TLFA) [19]. To alleviate the need for large training data, metric learning is adopted as an effective method in [17]. The main idea of TF is to learn a metric function with deep neural networks that is applicable to test classes that are not seen during training. However, TF is inferior in the formulation. First, it is not scalable to leverage large scale auxiliary data since the model training relies on complex sample pairs and an elaborate triplet search process. Besides, it is inferior in design as its model optimization is not explicitly designed for solving the target tasks. To address these above limitations, the seminal data augmentation strategy is investigated and tailored particularly for website fingerprinting data, including rotation, masking, and mixing [18]. Nonetheless, this method is ineffective for the extreme cases such as 1-shot due to lacking of sufficient knowledge about a new class. This weakness can be alleviated by a more recent TLFA method [19] that pre-trains a feature embedding model on a large auxiliary dataset for knowledge transfer. On the other hand, this method comes with two extra limitations: (1) assuming similar distributions for auxiliary and target datasets, which does not always hold, and (2) no mechanism to adapt the pre-trained feature embedding to a new task. Our method solves all these limitations with a more advanced and principled formulation – meta-learning.

2.3. Meta-learning

As the dominant few-shot learning approach, meta-learning has the capability of learning a new model with very few examples [42–45]. That is, it aims to learn a learner that can generalize well from old tasks to new tasks without class overlap across tasks.

Table 2
Summary of the main acronyms used across the entire paper.

Acronym	Full name and description
WF	Website fingerprinting
FS-WFA	few-shot website fingerprinting attack
N-way *K*-shot	*N* websites unseen before and *K* samples per new class can be used for training
CNN	Convolutional Neural Networks
CUMUL	Method proposed in the paper [12].
k-FP	k-fingerprinting, proposed in the paper [38].
DF	Deep Fingerprinting, proposed in the paper [15].
VGG	Very Deep Convolutional Networks, propose in the paper [37].
Var-CNN	Method proposed in the paper [16]
ResNet	Deep Residual Network, proposed in the paper [38]
GoogleNet	CNN model proposed in the paper [46]
ResNeXT	CNN model proposed in the paper [47]
TF	Triplet Fingerprinting, proposed in the paper [17].
TF*	TF trained by our improved data sampling strategy.
TLFA	Transfer Learning Fingerprinting Attack, proposed in the paper [19].
HDA	Harmonious Data Augmentation, proposed in the paper [18].
HDA + Var-CNN	Var-CNN trained by the data augmented by HDA.
miss Type I	A testing sample from one of the monitored sites is *incorrectly* predicted as unmonitored.
miss Type II	A testing sample from one of the monitored sites is *incorrectly* predicted as others.
FNR^2/FNR^m	Binary-class/Multi-class False Negative Rate
TFR^2/TPR^m	Binary-class/Multi-class True Positive Rate
ROC	Receiver Operating Characteristic curve
AUC^2/AUC^m	*Binary-class/Multi-class Area Under Curve*

Specifically, it performs the model optimization at the task level that simulates the test few-shot scenario. In such a way, the model is learned to adapt to a given new task rapidly with only a few labeled training samples. While meta-learning has been extensively explored in computer vision [43], how well it can address few-shot WF attack remains unclear and unexplored. In this work, we fill this gap by exploring meta-learning for few-shot WF attack for the first time. Our study shows that this is a new and promising direction that offers a superior solution over existing alternatives in terms of both training scalability and model performance.

To facilitate the reading and understanding, we provide an Acronyms list for the terminologies used in this paper (Table 2).

3. Method

3.1. Problem context

We start with the ordinary website fingerprinting attack problem. The common observations/instances are data traffic traces $\boldsymbol{x}$ produced by one visit to a website y. A machine learning algorithm aims to train a model that can predict automatically the website based on a traffic trace as input. Treating each website as a specific class, this is essentially a multi-class classification problem. For model training, a labeled training set $D_{train} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ is often provided, where $y_i \in Y = \{1, 2, \cdots, K\}$ specifying one of K target websites. For model testing, a labeled testing set $D_{test} = \{(\hat{\boldsymbol{x}}_j, \hat{y}_j)\}_{j=1}^{M}$ is also provided, where the ground-truth labels $\hat{y}_i \in Y$ are unseen to the model and used for performance evaluation. Whilst a number of increasingly stronger deep learning methods based on various Convolutional Neural Network (CNN) models (*e.g.*, VGG [37], GoogleNet [46], ResNet [38], ResNeXT [47]) have been proposed, they cannot fully respect the deployment requirements of real-world applications. One main reason is that they all make artificial assumptions that the training and testing classes are identical, and the user will not visit new websites during the deployment time. In practice, these conditions typically do not hold, and we often require the model to adapt to new attack tasks (*i.e.*, new websites/classes) rapidly with only *a handful of labeled training samples*. We call this problem as *few-shot website fingerprinting attack* (FS-WFA).

In FS-WFA, we consider a more realistic problem definition. Given a meta-test dataset D_{test}, we sample a N-way K-shot (N new websites each with K labeled training samples) classification task to test a learned model $\boldsymbol{\theta}$. To train the model $\boldsymbol{\theta}$ in a way that it can perform well on those previously unseen classification tasks, meta-training by episodic learning is an intuitive paradigm [17]. Concretely, a large number of N-way K-shot tasks are randomly sampled from a meta-training set D_{train}, and then used to train the model in an episodic manner. In each episode, we start with sampling N classes C from D_{train} at random, from which labeled training samples are then randomly drawn to create a *support* set $\mathcal{S}$ and a *query* set $\mathcal{Q}$ consisting of K and Q samples per class, respectively. Formally, the support and query sets are defined as:

$$\mathcal{S} = \{(x_i, y_i) \mid y_i \in C\}_{i=1}^{NK}, \tag{1}$$

$$\mathcal{Q} = \{(x_i, y_i) \mid y_i \in C\}_{i=1}^{NQ}. \tag{2}$$

Note, $\mathcal{S} \cap \mathcal{Q} = \phi$ are sample-wise non-overlapping.

Note that as the objective is to obtain a learner able to recognize novel classes each with only a few labeled examples, D_{train} and D_{test} are set to be disjoint in the class space, *i.e.*, $D_{\text{train}} \cap D_{\text{test}} = \phi$. Unlike the sparsely annotated meta-test classes, each meta-training class comes with abundant labeled training data that allow to sample a sufficient number of episodes for meta-training.

Limitations of existing methods. There are very few existing works [17–19] which also consider the FS-WFA problem setting. In particular, we study the same setting as TF [17] and TFLA [19]. TF has yielded encouraging results by taking a metric learning strategy. However, it has several fundamental limitations: (1) Relying on the triplets based metric learning, this method suffers from complex pair and triplet construction and more demanding training tricks such as hard pairs and triplets mining. This renders TF inefficient and ineffective to leverage large training data. (2) Critically, its model optimization does not match the scenarios of target tasks, leading to inferior model performance.

On the other hand, TFLA has improved the results over TF by utilizing the large-scale pre-training data to learn a stronger embedding model. Relying on extra data with a similar distribution as the target tasks, this method may suffer from new tasks with domain shift and different distributions, as its pre-trained embedding component is frozen during few-shot learning.

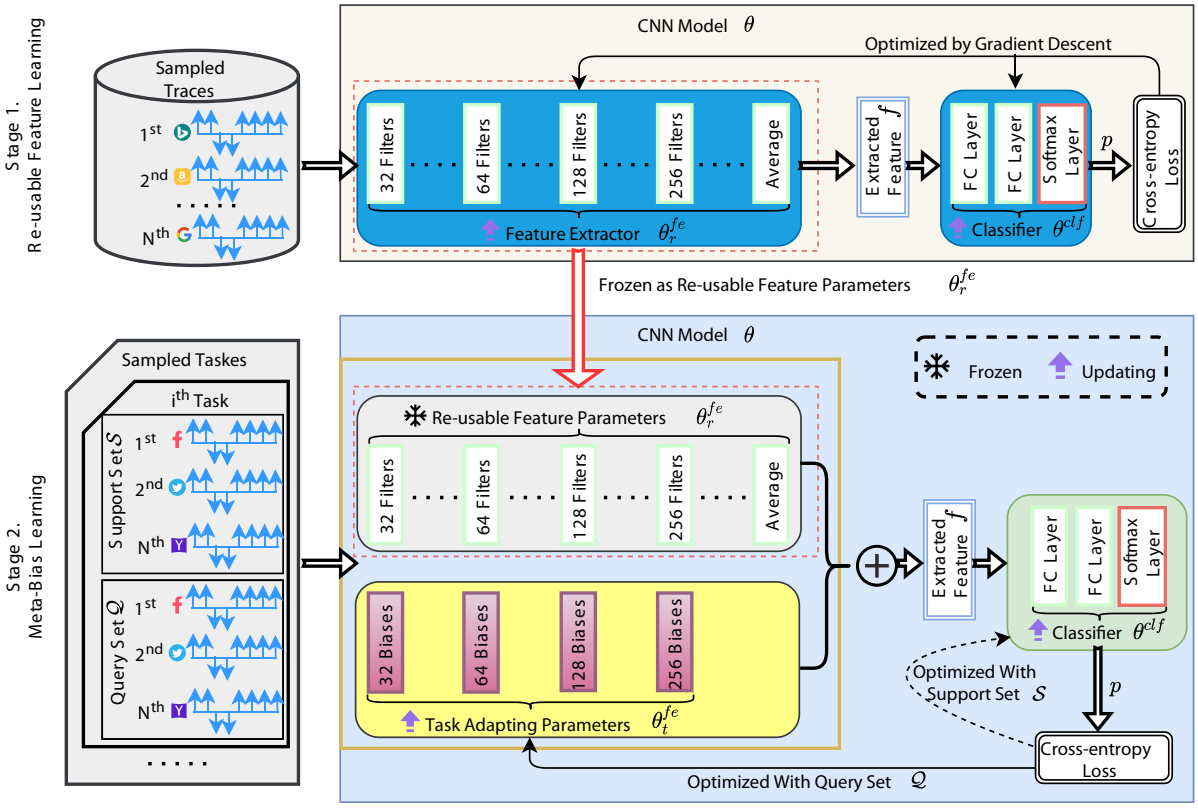

Fig. 1. Schematic overview of the proposed Meta-Bias Learning (MBL) model. MBL aims to address the fundamental limitation of existing methods in fast task adaptation with a few labeled training samples, *i.e.*, few-shot WF attack. It is based on meta-learning that simulates and optimizes the scenarios of test tasks in deep learning. Moreover, a novel parameter factorization idea is proposed to decompose a standard convolution operations of a CNN model θ into two parts functionally: re-usable feature parameters θ_r^{fe} and task adapting parameters θ_t^{fe}. To reduce the burden of task adaptation, θ_t^{fe} is designed to be the bias parameters in a small size (*e.g.*, 0.01%). Our MBL is trained in two stages. **(a)** In the first stage, we train the re-usable feature parameters θ_r^{fe} by supervised classification learning, along with the classifier component θ^{clf}. This choice is computationally efficient and more scalable to large training data as compared to pairwise loss based metric learning. **(b)** In the second stage, we meta-learn the task adapting parameters θ_t^{fe} by episodic training, which can be added to θ_r^{fe} to define task-adaptive feature representations, *i.e.*, task adaptation by meta-learning. As the classes of every episode are different, we need to fine-tune the classifier's parameters θ^{clf} for every individual task/episode.

3.2. Meta-bias learning

To overcome the aforementioned limitations of existing state-of-the-art method, we formulate a novel *Meta-Bias Learning* (MBL) model, as illustrated in Fig. 1. It is based on meta-learning that simulates and optimizes exactly the same scenario of test tasks. Moreover, the model parameters are factorized into two complementary components, one is optimized with common supervised learning and the other is meta-trained by episodic training. Notably, the latter takes only a very tiny fraction of parameters so that meta-training can be more effectively conducted with a reasonable training difficulty. This design can take advantage of supervised learning in extracting re-usable feature representations, whilst properly leveraging meta-learning's capability in task adaptation. By exploiting the computationally scalable supervised classification learning for the former component, our MBL addresses well all limitations of the state of the art TF concurrently. **Architecture.** We consider a deep learning architecture, in particular convolutional neural networks (CNN), that can learn feature representations from the raw training data and conduct WF attack via classification end-to-end [15]. This avoids complex hand-crafting feature engineering. Since the feature representations are fully parameterized, this provides us a favourable opportunity for principled feature factorization.

Raw input. For typical Tor networks, the raw representation $\boldsymbol{x}$ of a specific traffic trace consists of a sequence of temporally successive Tor cells travelling between a target user and a website visited. It is derived from TCP/IP data. After discarding retransmitted TCP/IP packets, TLS records are first reconstructed, their lengths are then rounded down to the nearest multiple of 512 to form the final sequence data $\boldsymbol{x}$. In specific data value, each $\boldsymbol{x}$ is a sequence of 1 (outgoing cell) and -1 (incoming cell), with a variable length. This raw representation is hence known as direction sample. Besides, temporal information about inter-packet time is another type of modality, but with high reliance on network conditions. It is therefore not stable even with more noises. We only consider the direction data in this study, which are more scalable and generic raw data representations.

Deep learning model. A CNN model θ consists of two parts: a feature encoder θ^{fe} and a classifier θ^{clf}. Specifically, the feature encoder θ^{fe} contains multiple stacked convolutional layers with non-linear activation functions such as ReLU [48], along with normalization and pooling operations. Our CNN has eight convolutional layers with 32/32/64/64/128/128/256/256 filters, respectively. A batch normalization layer is added after each convolutional layer. The learnable parameters of batch normalization include mean and variance. With convolutional operations, the filters of each layer transform input sequences using learnable parameters and output new feature sequences. We use a kernel size of 8 in each convolutional layer to capture local feature patterns. By stacking more layers with normalization and pooling operation, the model can perceive the information of larger regions and enables to capture and model wider context information across the time. The feature representations $\boldsymbol{f}$ of individual WF samples $\boldsymbol{x}$ are obtained as the output vectors of a global average pooling layer. Among these layers of the classifier θ^{clf}, the first FC layer is used

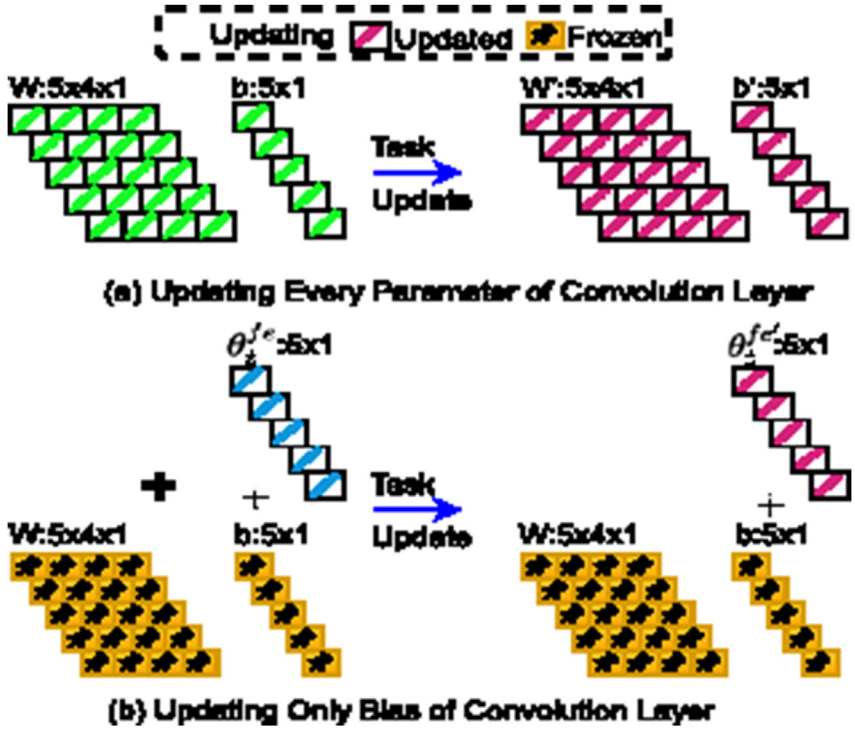

Fig. 2. Illustration of the proposed parameter factorization of convolution operations in comparison with the standard one. It is a case of a convolutional layer with five 1D kernels, each in the shape of 4×1. **(a)** For training the standard convolutional network, all the parameters including kernel weights (*e.g.*, W in shape of 5×4) and biases (b in shape of 5×1) are updated in each training iteration. **(b)** In our method, we factorize the convolution parameters by introducing an extra meta-bias θ_t^{fe} with the same dimension and number as the original bias b. Adding θ_t^{fe} to b is designed as task adaptation. This brings about two advantages: (1) No need to meta-train the whole parameter set but only a small number of meta-bias which is not only more efficient but also more effective. (2) The model can be pre-trained and then frozen to better leverage large-scale auxiliary data.

to project the feature $\boldsymbol{f}$ to a latent embedding space, which aims to distil more compact semantic feature representations. We denote the whole parameters of the CNN model as $\theta = [\theta^{fe}, \theta^{clf}]$.

Parameter factorization. The CNN model MBL, as described above is a standard formulation. To improve the efficacy of meta-learning, we propose a novel parameter factorization idea. Formally, we factorize the whole parameters θ of a CNN model(Fig. 2(a)) into two parts: the bias of convolutions as task adapting feature parameters θ_t and the rest of convolutions as re-usable feature parameters θ_r (Fig. 2(b)). This process is formulated as:

$$\theta^{fe} \xrightarrow[factorizing]{} \theta_t^{fe}(\text{task adapting}) + \theta_r^{fe}(\text{re-usable}) \tag{3}$$

Such a factorization is designed to have a small number of task adapting parameters that need to be meta-learned. In our case, the size of θ_t^{fe} takes only 0.01% of the whole parameters θ^{fe}. Since the input has only one channel, the number of task-adaptive parameters is (32 + 64 + 128 + 256) * 2 = 960. Given the parameter number of feature extractor as $(1 \times 32 + 32 \times 32 + 32 \times 64 + 64 \times 64 + 64 \times 128 + 128 \times 128 + 128 \times 256 + 256 \times 256) \times 8 + 960 + 960 \times 2 = 1043520$. So the ratio of task-adaptive parameters with respective the feature extractor is:

$$\frac{960}{1043520} \approx 0.1\% \tag{4}$$

This allows to better leverage meta-training.

Model training. Our MBL is trained in two stages:

1. In the first stage, we train a CNN network for optimizing re-usable feature parameters θ_r^{fe} along with the classifier parameters θ^{clf}.
2. In the second stage, we meta-optimize task adapting parameters θ_t by episodic training, and combine θ_t^{fe} with θ_r^{fe} and θ^{clf} to define task-specific models.

Stage 1: Re-usable feature learning. This stage of training follows the standard supervised classification learning procedure. Specifically, we start with a randomly initialized CNN model. To optimize it, we adopt the stochastic gradient descent algorithm for model parameter update at every iteration t as:

$$\theta(t+1) =: \theta(t) - \alpha \nabla \mathcal{L}_{\mathcal{D}}(\theta) \tag{5}$$

where $\mathcal{L}$ denotes an empirical loss of the current mini-batch $\mathcal{D}$ and ∇ specifies the gradient function of the loss with respect to the model parameters. The hyper-parameter α is known as the learning rate. As we have ground-truth labels for every training sample, we utilize the softmax based cross-entropy loss function as:

$$\mathcal{L}_{\mathcal{D}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} l(\boldsymbol{p}, y) \tag{6}$$

$$= \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} -\log \boldsymbol{p}(y) \tag{7}$$

where $\boldsymbol{p}$ is the predicted class distribution vector for a training sample x with ground-truth label y, and $\boldsymbol{p}(y)$ takes the y-th probability value. The objective of this stage is to optimize re-usable feature parameters θ_r^{fe} which will be frozen once trained in the subsequent process. As supervised classification learning is much simpler and more scalable than triplet based metric learning as used in TF [17], stronger re-usable feature representations can be obtained. **Stage 2: Meta-Bias Learning.** This stage takes the meta-learning strategy based on episodic training. The objective is to enhance the re-usable feature model θ_r^{fe} with task adapting capability essential to FS-WFA. Under the proposed idea, we meta-train

the task adapting parameters θ_t^{fe} (*i.e.*, meta bias). This is done by learning from a large number of randomly sampled tasks/episodes from the same training data as in Stage 1.

We design a two-step meta-training algorithm. Given a sampled task (episode) $E = \{\mathcal{S}, \mathcal{Q}\}$ including a support set $\mathcal{S}$ and a query set $\mathcal{Q}$, the task-specific classifier θ^{clf} is first optimized by the loss of $\mathcal{S}$ by gradient descent:

$$\theta^{clf}(t+1) \leftarrow \theta^{clf}(t) - \beta \nabla \mathcal{L}_{\mathcal{S}}\left(\theta^{clf} | \theta_r^{fe}, \theta_t^{fe}\right) \tag{8}$$

where β is the learning rate, θ_r^{fe} and θ_t^{fe} are frozen. It only updates the parameters θ^{clf} in the first step. This is because the classes of each episode are different from each other, and the classifier needs to update at the beginning.

In the second step, we then optimize the bias parameters θ_t^{fe} with the query set $\mathcal{Q}$ given the frozen classifier and re-usable feature parameters. This is done similarly by gradient descent as:

$$\theta_t^{fe}(t+1) \leftarrow \theta_t^{fe}(t) - \gamma \nabla \mathcal{L}_{\mathcal{Q}}\left(\theta_t^{fe} | \theta^{clf}, \theta_r^{fe}\right) \tag{9}$$

where θ_t^{fe} is initialized as zero at the beginning of meta-training *once*. Note, this is different from θ^{clf} which needs to be initialized randomly and independently for every task.

To integrate θ_t^{fe} into the CNN model, we respect the standard convolutional operations for computational compatibility on modern hardware platforms by considering the proposed meta bias as the residual of original bias. Suppose the l-th layer of our CNN model has K convolutional kernels, denoted as F^l. Then, there will be K meta bias parameters $\theta_{t,l}^{fe}$ corresponding to F^l. For a given input feature $\boldsymbol{f}^l$, the two parts are integrated together to conduct a convolution operation in the following manner:

$$\boldsymbol{f}^{l+1} = conv(F^l, \boldsymbol{f}^l) + \theta_{t,l}^{fe} \tag{10}$$

where $conv()$ denotes the standard convolutional operation as illustrated in Fig. 2. This design brings a few significant advantages: (1) As the re-usable feature parameters are frozen, *catastrophic forgetting* can be avoided in meta-training. (2) The meta-training overload is tiny, so the optimization challenge is mitigated.

Algorithm Summary. To summarize all details of our MBL method, we present an algorithm routine (Algorithm 1) that covers the key steps of model training.

Algorithm 1 Meta-Bias Learning.

Input: Training dataset: D_{train};

Output: A FS-WFA CNN model with the re-usable feature $\theta_r^{fe}$and meta-bias θ_t^{fe} parameters optimized./* *Stage 1: supervised classification learning* */

```
Randomly initialize θ_r^fe and θ^clf;
for Samples in D do
    Calculate L_D by Eq. (6);
    Optimize θ_r^fe and θ^clf by Eq. (5);
end for/* Stage 2: meta-training with episodes */
Initialize θ_t^fe by zeros;
Reset θ^clf for the classes of a training episode;
for each iteration in meta-training do
    Randomly sample a task/episode from D_train;
    Format the episode into a support S and query Q set;
    /* Step 1: adapt the classifier */
    Randomly initialize θ^clf;
    for Sample in S do
        Calculate L_S;
        Optimize θ^clf by Eq. (8);
    end for/* Step 2: adapt the meta-bias and classifier jointly */
    Optimize θ_t^fe by Eq. (9);
end for
```

3.3. Model testing

In model testing, given a new task with a few labeled training samples per class/website, we aim to build a task-specific model. Specifically, we *freeze* the re-usable feature θ_r^{fe}, and fine-tune the parameters θ^{clf} of the classifier followed by optimizing the meta bias θ_t^{fe}. In this way, the overall feature extractor is optimized particularly for a specific task via the fine-tuning of θ_t^{fe}. Since then, the whole model is ready to accept testing samples of the current new task for FS-WFA.

4. Experiments

4.1. Datasets and protocols

We evaluate our MBL on three standard WF attack datasets in both closed-world and open-world scenarios. For *closed-world* attack, all test samples are assumed to belong to one of the training *(monitored)* classes/websites. For *open-world* attack, however, the above assumption is eliminated, *i.e.*, a test sample may be produced by a *non-target* (unmonitored) website. The latter is more realistic yet more challenging as identifying if a test sample is in one of the target classes or not is non-trivial.

(1) **AWF dataset** [14]: We use its largest dataset containing a total of 900 monitored target websites, each with 2500 raw feature traces. It was collected with Tor Browser 8.5a7 on Tor 0.2.8.11. All 900 websites are divided into a split of 576/144/180 for meta-training (AWF^{tr}), meta-validation (AWF^{va}) and meta-test (AWF^{te}), respectively.

Interestingly, it also provides timestamp data that can be used for evaluating the distribution drift problem during test. Since the end of the initial data collection, after a period of $g \in \{3, 10, 14, 28, 42\}$ days, an independent set of 100 samples for each website was collected. These five extra sets allow to evaluate the robustness and stability of any test algorithm against the distribution shift introduced over time. For easy reference, we denote them in the form of AWF_{180}^{g} where g refers to the time gap in data collection. These test sets are used for closed-world evaluation.

For open-world evaluation, another set of 400K samples $\text{AWFUM}_{400\text{K}}$ from unmonitored websites can be used. Each of these samples was generated by a visit to a page of the top 400K of Alexa websites. To enrich the test setting, we randomly select samples from $\text{AWFUM}_{400\text{K}}$ as the distracting data.

(2) **DS-19**$_{100}$ [24]: This is the latest dataset collected with Tor Browser 8.5a7 on Tor 0.4.0.1-alpha in 2019, containing 100 homepages of Alexa top 100 websites. Each homepage has 100 instances. Compared with AWF, this dataset is smaller but more up-to-date. All the websites are divided into the splits of 60/20/20 for meta-training ($\text{DS-19}_{100}^{\text{tr}}$), meta-validation ($\text{DS-19}_{100}^{\text{val}}$) and meta-test ($\text{DS-19}_{100}^{\text{te}}$), respectively. Furthermore, we apply two defense approaches, WTF-PAD [49] and FRONT [24], to defense the dataset DS-19. These results in $\text{DS-19}_{100,pad}$ for the WTF-PAD defense and $\text{DS-19}_{100,front}$ for the stronger FRONT defense. Similarly as DS-19_{100}, $\text{DS-19}_{100,pad}$ is divided into $\text{DS-19}_{100,pad}^{\text{tr}}$ for meta-training, $\text{DS-19}_{100,pad}^{\text{val}}$ for meta-validation and $\text{DS-19}_{100,pad}^{\text{te}}$ for meta-test. The same splitting is applied to $\text{DS-19}_{100,front}$.

4.2. Evaluation metrics

For the closed-world setting, we use accuracy, precision, recall and F1 score to measure the effectiveness of the attacker, which is the proportion of monitored traces correctly identified. Formally, they are defined as:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}(Predict(T_i) == Label(T_i)) \tag{11}$$

$$Precision = \frac{1}{L_m}\sum_{l=1}^{L_m}\frac{\sum_{i=1}^{N}(Label(T_i)==l, Predict(T_i)==l))}{\sum_{i=1}^{N}(Predict(T_i)==l)} \quad (12)$$

$$Recall = \frac{1}{L_m}\sum_{l=1}^{L_m}\frac{\sum_{i=1}^{N}(Label(T_i)==l, Predict(T_i)==l)}{\sum_{i=1}^{N}(Label(T_i)==l)} \quad (13)$$

$$F_1 = 2\cdot\frac{Precision\cdot Recall}{Precision + Recall} \quad (14)$$

where T_i is the i-th testing trace, N is the number of total testing traces, l is the l-th monitored website label, L_m is the total number of monitored websites, $Prediction(T_i)$ is the predicted label of T_i, $Label(T_i)$ is the ground-truth label of T_i.

In the open-world setting, we do not assume that all test traces are from monitored websites. Instead, most traces captured by an attacker may be generated by unknown/unmonitored websites. Accuracy rate hence cannot measure this property.

For open-world setting evaluation, we must quantify three error types[50]. The first type is false alarm — a trace from an unmonitored website is *incorrectly* recognized as one monitored website. This error is quantified by *False Positive Rate* (FPR) defined as:

$$FPR = \frac{M_{um}}{N_{um}} \quad (15)$$

where N^{um} is the number of all testing traces D_{um} generated by unmonitored websites, and M^{um} is the number of testing traces in D_{um} but are *incorrectly* recognized as monitored ones.

The second type is *miss Type I* — a testing sample from one of the monitored sites D_m is *incorrectly* predicted as unmonitored. We quantify this by *Binary-class False Negative Rate* (FNR^2) defined as:

$$FNR^2 = \frac{M_m}{N_m} \quad (16)$$

where $N_m = |D_m|$ is the total number of testing traces generated by monitored websites, and M_m is the number of samples in D_m but *mistakenly* recognized as unmonitored ones.

The third type is *miss Type II* — a testing trace x_i from D_m with the class label y_i is *incorrectly* predicted as another monitored website class y_j, $i \neq j$, or even as unmonitored ones. We denote these samples collectively as D_m^{fm}. Hence this is a more strict measurement as compared to the first type. We quantify this by *Multi-class False Negative Rate* (FNR^m) defined as:

$$FNR^m = \frac{M'_m}{N_m} \quad (17)$$

where $M'_m = |D_m^{fm}|$.

In practice, a more intuitive measure may be *True Positive Rate* (TPR):

$$TPR^2 = 1 - FNR^2 \quad (18)$$

$$TPR^m = 1 - FNR^m \quad (19)$$

In the following open-world experiments, we adopt both TPR@FPR measures for performance evaluation. To yield more compact overall results, we generate Receiver Operating Characteristic (ROC) curves of TPR-vs-FPR and use two Area Under Curve (AUC) scores: *Binary-class Area Under Curve* (AUC^2) focusing on *miss Type I*, and *Multi-class Area Under Curve* (AUC^m) focusing on *miss Type II*. The latter is more difficult and more strict. The formulations of AUC^2 and AUC^m are written as follows:

$$AUC^2 = \int_{FPR^2=0}^{1} TPR^2 \mathrm{d}\left(FPR^2\right) \quad (20)$$

$$AUC^m = \int_{FPR^m=0}^{1} TPR^m \mathrm{d}(FPR^m) \quad (21)$$

4.3. experimental setup

Network architecture. Our CNN model contains four blocks, each having four convolutional layers with 8×1 kernels. Each convolutional layer is followed by an 8×1 max-pooling layer and a batch-norm layer. The number of convolutional kernels/filters starts from 32 and is doubled from one block to the next. The first block uses the ELU activate function, whilst the remaining three blocks use the ReLU activate function. Following four blocks is a mean-pooing layer to compress the output feature maps to a feature embedding vector. For other hyper-parameters, we follow the same setting as TF [15] for fair comparisons. **Supervised classification learning.** We adopt the SGD optimizer to train the re-usable feature parameters. The learning rate is initialized as 0.1 and decays by 0.2 every 30 epochs. We set the dropout rate as 0.1 and batch size as 512. We train a total of 110 epochs. **Meta-training setup.** For AWF, we consider the 100-class/website WF attack setting. During meta-training, from each class (*i.e.*, website), we randomly sample $K(\in \{1, 5, 10, 15, 20\})$ training samples to the support set and 15 training samples to the query set for every episode/task. This forms the K-shot cases in the few-shot learning WF attack setting. The same process applies to meta-validation and meta-test. For the smaller DF datasets (DS-19_{100}, $\text{DS-19}_{100,pad}$, $\text{DS-19}_{100,front}$), we consider the 1-shot, 2-shot, 3-shot, 4-shot and 5-shot WF attack settings using 10 classes. The task-specific classifier θ^{clf} is optimized by batch gradient descent with the learning rate of 0.01. It is updated with 100 iterations. The meta-bias θ_t^{fe} is trained by Adam optimizer, with the learning rate initialized as 0.0001 and decayed by 0.5 every 100 iterations. For all datasets, we sample 8000 random tasks for meta-training, 10 tasks for meta-validation, and 600 tasks for meta-test. Meta-validation performance is used to select the final model for meta-test.

4.4. Closed-world few-shot website fingerprint attack

Setting. We use the classification accuracy, precision, recall and F1 score as performance metric. We compare with the state-of-the-art few-shot learning WF method, Triplet Fingerprinting (TF) [17], TFLA [19], two conventional methods (CUMUL [12], k-FP [11]) and Var-CNN [16] which is trained by Harmonious Data Augmentation [18] (named as Var-CNN+HDA). In particular, the original TF was trained with 25 examples of each training website due to GPU memory constraints. For a fair comparison, we select 25 random samples for each epoch to supply it with more information whilst still keep it trainable on a normal machine. We find this data sampling strategy can improve TF's performance. To differentiate from the original TF model, we denote this improved variant as **TF***. For TLFA, we select its best classifier (Support Vector Machine) and embedding vector (type I, the output vector of the feature extractor). For AWF dataset, we test all the above algorithms. For DS-19, we focus on comparing the top-2 existing methods: TF and TLFA.

Results. The accuracy results of different methods on the AWF dataset are compared in Table 3. Notably, we omit the results of precision, recall and F1 score as they are similar in comparison. We have the following observations:

1. Even though rich domain knowledge is leveraged, hand-crafted feature based methods (CUMUL and k-FP) are not competitive anymore, no matter how few samples per class are available.
2. Few-shot learning methods (TF, TLFA and our MBL) are clearly superior suggesting the generic benefits of more effective knowledge transfer from training classes to test classes.

Table 3
Results of *closed-world* few-shot WF attack on AWF. *: Trained by our improved data sampling strategy. Metric: Accuracy.

Method	1-shot	5-shot	10-shot	15-shot	20-shot
CUMUL [12]	42.1 ± 5.5	72.2 ± 1.7	79.7 ± 1.4	83.3 ± 2.0	85.9 ± 0.6
k-FP [11]	36.6 ± 1.6	79.3 ± 1.0	83.9 ± 1.0	85.9 ± 0.6	87.5 ± 0.8
DF [15]	$\leqslant 10$	$\leqslant 10$	$\leqslant 10$	37.3 ± 10.0	70.0 ± 4.4
Var-CNN [16]	$\leqslant 10$	17.9 ± 1.5	41.4 ± 4.0	65.6 ± 1.9	78.7 ± 1.5
TF [17]	79.2 ± 1.3	92.2 ± 0.6	93.9 ± 0.2	94.4 ± 0.3	94.5 ± 0.2
TF* [17]	81.1 ± 0	92.6 ± 0	93.7 ± 0	94.5 ± 0	94.1 ± 0
TLFA [19]	89.3 ± 0	$\mathbf{97.3 \pm 0}$	$\mathbf{98.5 \pm 0}$	$\mathbf{98.6 \pm 0}$	$\mathbf{98.8 \pm 0}$
Var-CNN+HDA[18]	$\leqslant 10$	59.7 ± 1.5	74.7 ± 2.6	86.4 ± 1.3	90.7 ± 0.8
MBL	$\mathbf{92.5 \pm 0}$	97.2 ± 0	98.3 ± 0	97.9 ± 0	98.4 ± 0

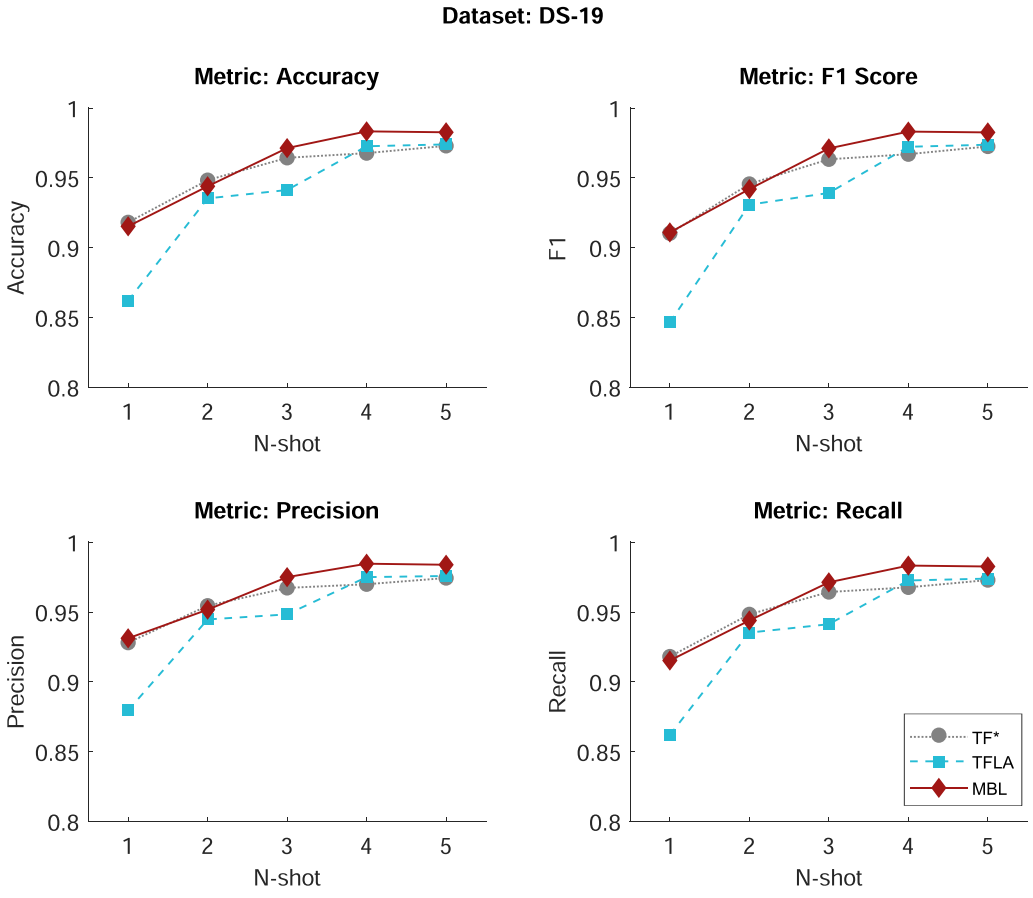

Fig. 3. *Closed-world* few-shot WF attack performance of TF*, TLFA and MBL on the DS-19 dataset. Metric: Accuracy, F1 score, Precision and Recall.

3. Without knowledge transfer from other websites, data augmentation (HDA) is reasonably effective only when a good number of (*e.g.*, 10+) training examples or shots are available.
4. Our model MBL is consistently superior to all other competitors in the most difficult 1-shot case. For example, our MBL achieves an accuracy margin of 3.2% over the best previous competitor TLFA, and 50.4% over CUMUL (the best traditional WF attack method for the 1-shot setting). When more shots are given, our MBL is competitive with the state-of-the-art method TLFA, whilst outperforming all the other competitors by a large margin. We stress that in many real-world applications, often only as few as one training sample is available and hence our MBL will be the best performer in those circumstances.
5. The performance deviation of pre-training based methods (TF, TLFA, and our model MBL) are all the least, suggesting their strong stability.

We further compare our MLB with top-2 methods (TLFA and TF*) on the DS-19 dataset in Fig. 3. To complement Table 3, we report the metrics of accuracy, F1 score, precision, and recall here. The main observations are that, (1) our MBL can now outperform both competitors in most cases, and (2) TF turns out to be better than TFLA. This suggests that our model MBL has superior robustness across datasets. In contrast, the performance advantage of TF and TLFA is inconsistent. Hence our method has generalization advantage to different applications.

4.5. Closed-world few-shot website fingerprint attack with time shift

Setting. We evaluate the closed-world few-shot WF attack scenario with varying time gaps between training data collection and testing data collection. We compare our MBL with the state-of-the-art TF [17] and TLFA [19]. **Results.** The comparative results in F1

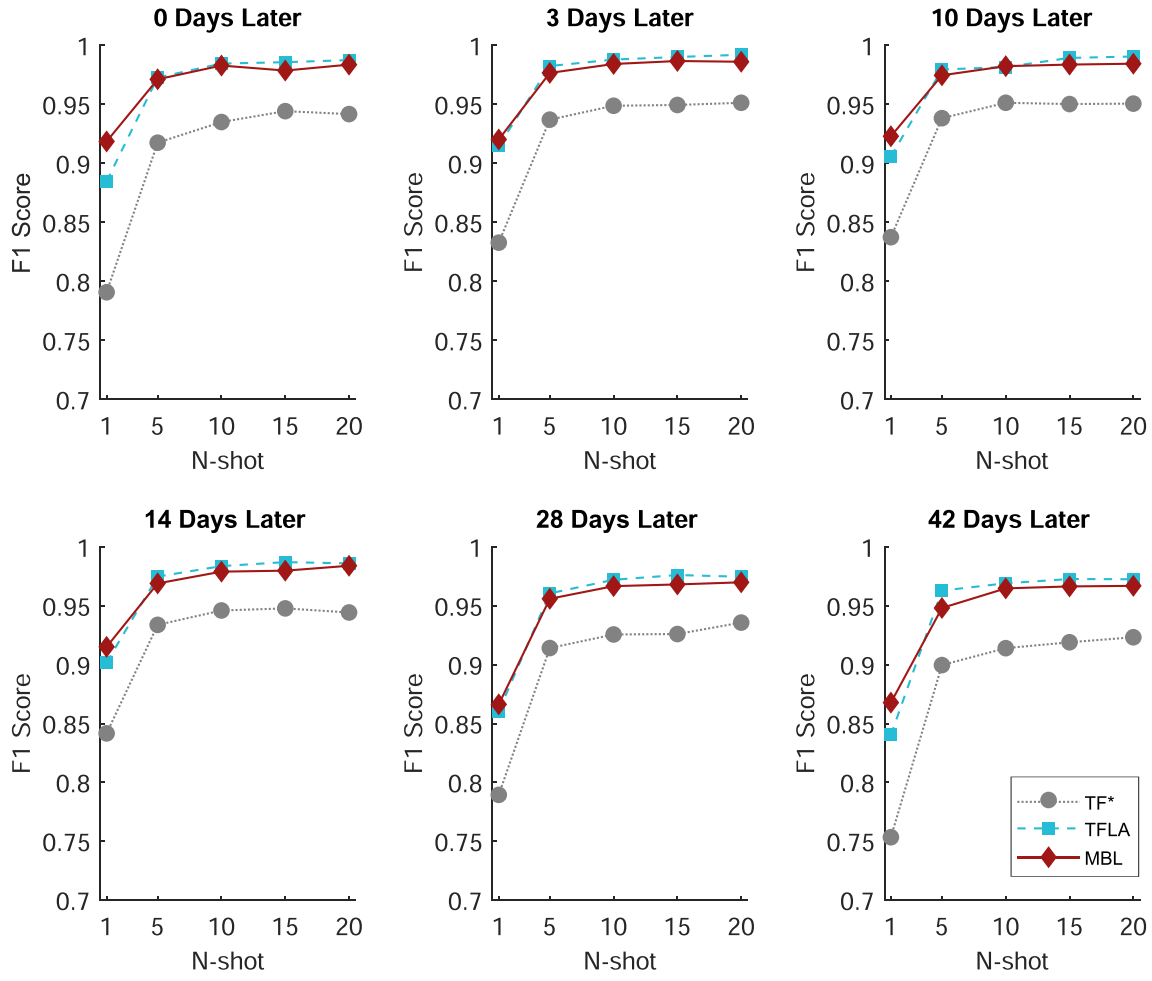

Fig. 4. *Closed-world* few-shot WF attack performance of TF*, TLFA and MBL over different train-test data time gaps (0–42 days). Dataset: AWF. Metric: F1 score.

score are shown in Fig. 4. The comparisons in accuracy, precision and recall are similar and hence omitted. We make the following observations:

1. Regardless of how large the time gap is, our model MBLsurpasses TF and TLFA consistently, with a visible margin for 1-shot cases across different day gaps. This suggests the consistent advantage of our MBL over varying data changes in time.
2. The performance drop is generally linear with the time gap, but TF, TLFA and MBL are all reasonably stable in performance. This suggests a good generality of deep pre-training learning methods. For instance, given a large 6-week time gap between training data collection and test data collection, MBL can still correctly deanonymize 96.7% out of 100 website visits. Interestingly, we also observe some improvements in cases of 3/10-days gap in comparison to the no-gap case. This may be caused by other co-occurring factors in data collection.
3. When more shots are provided, our MBL closely matches the state-of-the-art TLFA, whilst significantly surpassing TF by a large margin. This is similar to the case of no time gap in Table 3, suggesting the comparative stability among these methods.

4.6. Open-World few-shot website fingerprint attack

Setting. We conduct the open-world WF attack experiment in a few-shot learning setting to test the more realistic performance of our model MBL. In each task, we randomly select 100 websites from AWF^{te} as target (monitored) classes and those of AWFUM as non-target (unmonitored) classes. As discussed earlier, a few labeled samples in support set are used to adapt the model. For the query set, we select 15 samples from each target class and 9K-400K samples from AWFUM as distracting data typically encountered in real-world use. We adopt the proposed AUC metrics to measure the performances of our MBL, TF [15], and TLFA [19]. **Results.** The comparative results are reported in Table 4 and Fig. 5. We have the following observations.

1. Our MBL outperforms TF consistently across all different shot cases and on both AUC metrics. Although there exist marginal disadvantages against TLFA in the 10/15/20-shot cases, our MBL is more robust because it always keeps high performance even when only one sample per website is available. However, both TF and TLFA collapse in the 1-shot case. This consistently suggests the performance and stability advantages of our Meta-Bias Learning idea.
2. The results of AUC^m is usually lower than that of AUC^2, as expected. This is because the former is a more strict metric requiring the model to predict the correct target class labels whereas the latter does not.
3. All methods perform similarly across different size of unmonitored websites. This indicates their stability against the distracting classes in operation.
4. MBL can better leverage fewer labeled training samples than TF and TLFA, suggesting higher data efficacy. This is potentially due to the reduction of parameters in meta-learning.

Table 4
Results of *open-world* few-shot website fingerprint attack. Metrics: AUC based on binary-class (AUC^2) and multi-class (AUC^m) true positive rate and false positive rate. #unmonitored denotes the number of unmonitored websites. *: Trained by our improved data sampling strategy.

#unmonitored	Method	1-shot		5-shot		10-shot		15-shot		20-shot	
		AUC^2	AUC^m	AUC^2	AUC^m	AUC^2	AUC^m	AUC^2	AUC^m	AUC^2	AUC^m
9k	TF* [17]	0.500	0.005	0.927	0.888	0.940	0.901	0.945	0.911	0.947	0.913
	TLFA [19]	0.519	0.005	0.940	0.913	**0.964**	**0.951**	0.970	0.958	0.973	**0.965**
	MBL	**0.895**	**0.840**	**0.950**	**0.934**	0.958	0.946	0.960	0.950	0.962	0.952
50k	TF* [17]	0.500	0.005	0.929	0.886	0.939	0.905	0.942	0.910	0.946	0.912
	TLFA [19]	0.540	0.005	0.943	0.917	**0.968**	**0.953**	**0.974**	**0.963**	**0.976**	**0.968**
	MBL	**0.900**	**0.844**	**0.954**	**0.938**	0.961	0.950	0.964	0.953	0.966	0.956
100k	TF* [17]	0.500	0.005	0.924	0.882	0.936	0.897	0.941	0.908	0.943	0.910
	TLFA [19]	0.546	0.005	0.937	0.908	**0.967**	**0.953**	**0.973**	**0.961**	**0.976**	**0.967**
	MBL	**0.901**	**0.846**	**0.955**	**0.939**	0.961	0.950	0.965	0.954	0.965	0.956
200k	TF* [17]	0.500	0.005	0.926	0.885	0.935	0.898	0.939	0.904	0.942	0.907
	TLFA [19]	0.545	0.005	0.944	0.918	**0.968**	**0.954**	**0.972**	**0.959**	**0.976**	**0.967**
	MBL	**0.900**	**0.844**	**0.955**	**0.938**	0.963	0.950	0.965	0.954	0.966	0.956
400k	TF* [17]	0.500	0.005	0.924	0.883	0.929	0.888	0.932	0.892	0.940	0.902
	TLFA [19]	0.557	0.005	0.948	0.923	**0.965**	**0.950**	**0.974**	**0.964**	**0.975**	**0.967**
	MBL	**0.904**	**0.846**	**0.956**	**0.940**	0.963	0.950	0.965	0.954	0.966	0.956

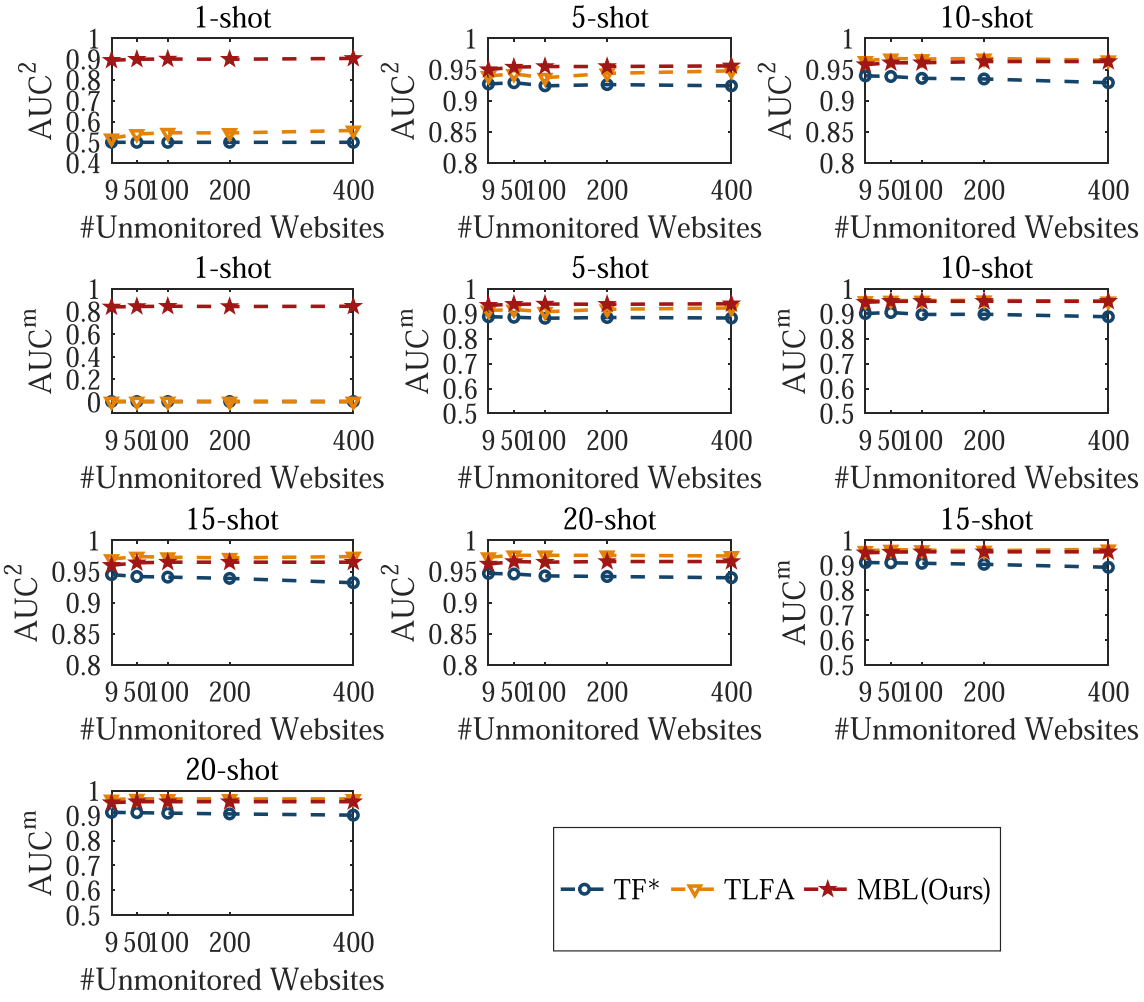

Fig. 5. Results of *open-world* few-shot website fingerprint attack. Metrics: AUC in binary-class (AUC^2) and multi-class (AUC^m). *: Trained by our improved data sampling strategy..

4.7. Few-shot website fingerprint attack against defense

Setting. We further test a more challenging few-shot website fingerprint attack scenario with defenses involved in the closed-world scenario. We consider two defenses: WTF-PAD [51] which is the most popular defense in Tor, and FRONT [24] which is the latest zero-delay defense. The datasets DS-$19_{100,pad}$ and DS-$19_{100,front}$ are used for this experiment. We adopt the 10-way 1/2/3/4/5-shot test settings. We use the state-of-the-art TF [17]) and TLFA [19] for comparison. **Results.** We report the results of closed-world WF at-

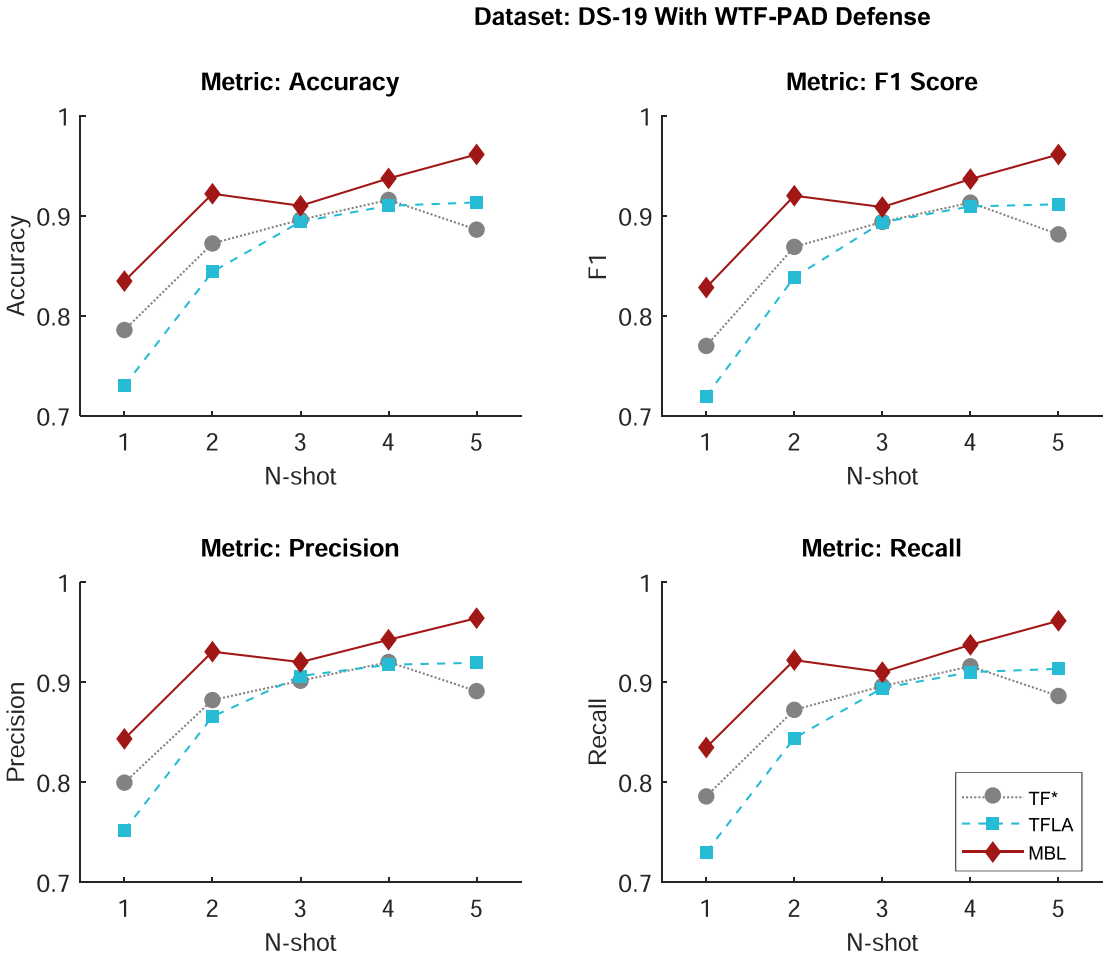

Fig. 6. *Closed-world* few-shot WF attack performance of TF*, TLFA and MBL on the DS-19 dataset with the WTF-PAD defense. Metric: Accuracy, F1 score, precision and recall.

Table 5
Closed-world few-shot WF attack performance of TF*, TFLA, MBL over the DS-19 dataset with the FRONT defense. Metric: Accuracy, F1 score, precision and recall.

Metric	Method	1-shot	2-shot	3-shot	4-shot	5-shot
Accuracy	TF* [17]	66.0 ± 0.1	77.4 ± 0	80.3 ± 0	82.2 ± 0	80.8 ± 0
	TLFA [19]	69.1 ± 0.1	80.0 ± 0	83.5 ± 0	84.3 ± 0	87.2 ± 0
	MBL	**80.5 ± 0.1**	**86.7 ± 0**	**87.8 ± 0.1**	**90.8 ± 0**	**93.0 ± 0**
F1 Score	TF* [17]	65.2 ± 0.1	76.9 ± 0	80.2 ± 0	82.0 ± 0	80.7 ± 0
	TLFA [19]	67.5 ± 0.1	79.4 ± 0	83.3 ± 0	84.2 ± 0	87.1 ± 0
	MBL	**80.2 ± 0.1**	**86.3 ± 0**	**87.4 ± 0.1**	**90.7 ± 0**	**92.9 ± 0**
Precision	TF* [17]	67.7 ± 0.1	78.6 ± 0	81.6 ± 0	82.8 ± 0	81.9 ± 0
	TLFA [19]	71.4 ± 0.1	81.2 ± 0	85.6 ± 0	85.6 ± 0	88.4 ± 0
	MBL	**83.4 ± 0.1**	**88.1 ± 0**	**89.6 ± 0.1**	**91.9 ± 0**	**93.9 ± 0**
Recall	TF* [17]	66.0 ± 0.1	77.4 ± 0	80.3 ± 0	82.2 ± 0	80.8 ± 0
	TLFA [19]	69.1 ± 0.1	79.9 ± 0	83.5 ± 0	84.3 ± 0	87.3 ± 0
	MBL	**80.5 ± 0.1**	**86.7 ± 0**	**87.8 ± 0.1**	**90.8 ± 0**	**93.0 ± 0**

tack under WTF-PAD based defense in Fig. 6, and under FRONT based defense in Table 5. We make the following observations.

1. Under the WTF-PAD defense, the results in all the metrics (accuracy, F1 score, precision and recall) of every few-shot WF attack method decrease obviously, as compared to the no-defense case in Fig. 3. However, it is important to note that our model MBL now always performs well than its competitors in all the cases.
2. Our model MBLperforms best, yielding a big margin of 12.7% over TLFA and 15% over TF in the most difficult 1-shot case with FRONT defense. Given more training samples (2/3/4/5-shots), the superiority of MBL remains with a margin of 5.8% over TLFA and 12.2% over TF. This suggests that our method can generalize the advantages to more challenging test scenarios.
3. Our model is also more robust against both defense cases. In comparison, TF and TLFA only perform reasonably well under one particular defense whilst suffering more from the other defense. This indicates that our method is more general with wider applicability.

4.8. Ablation study

In this section, we investigate the impact of data augmentation on our model MBL. We consider two data augmentation methods during training: random rotation and random mask, as introduced

Table 6
Effect of data augmentation on *closed-world* few-shot WF attack performance. Dataset: DS-19. Metric: Accuracy, F1 score, Precision, and Recall.

Metric	Data Augmentation	1-shot	5-shot	10-shot	15-shot	20-shot
Accuracy	×	**91.5 ± 0**	**94.4 ± 0**	**97.1 ± 0**	**98.3 ± 0**	**98.3 ± 0**
	✓	86.3 ± 0.1	91.7 ± 0	90.7 ± 0.1	93.4 ± 0	95.8 ± 0
F1 Score	×	**91.1 ± 0.1**	**94.2 ± 0**	**97.1 ± 0**	**98.3 ± 0**	**98.4 ± 0**
	✓	85.5 ± 0.1	91.4 ± 0	90.3 ± 0.1	93.2 ± 0	95.8 ± 0
Precision	×	**93.1 ± 0.1**	**95.2 ± 0**	**97.5 ± 0**	**98.5 ± 0**	**98.4 ± 0**
	✓	88.4 ± 0.1	92.6 ± 0	91.7 ± 0.1	93.8 ± 0	96.4 ± 0
Recall	×	**91.5 ± 0**	**94.4 ± 0**	**97.1 ± 0**	**98.3 ± 0**	**98.3 ± 0**
	✓	86.3 ± 0.1	91.7 ± 0	90.7 ± 0.1	93.4 ± 0	95.8 ± 0

in HDA [18]. We apply the same optimized augmentation parameters. Specifically, we rotate the trace entirely forward or backward up to 20 steps, and mask out some sub-trace at a maximal length of 180 with zero.

Setting. We conduct this test on DS-19. We use the classification accuracy, F1 score, precision and recall as the performance metrics.

Results. The results are compared in Table 6. We have the following observations:

1. For all the metrics, MBL cannot benefit from the data augmentation. This is somewhat surprising. One plausible reason is that our Meta-Bias Learning is already sufficient to overcome the small training data challenge, and data perturbation may introduce some undesired and incompatible effects instead.
2. Often, we see that data augmentation also enlarges the performance deviation, leading to less stable model generalization. This implies that meta-learning may require special data augmentation to be effective.

5. Conclusion

In this work we have presented a novel *Meta-Bias Learning* (MBL) method for few-shot website fingerprinting attack. Compared with the dominant supervised learning based WF attack methods, few-shot learning based WF attack is a more practical useful yet under-studied problem. This removes the conventional, artificial assumption on the availability of large training data for every target website. In practice only a handful of training samples per website can be feasibly collected given the high dynamics of internet networks, frequent updates of websites, continual change of monitored sites and attack requirements over time. To that end, we explore the more advanced meta-learning approach in deep learning, a new research direction with vast potentials for scalable WF attack. In contrast to the previous solutions, our meta-learning method is more scalable in model training in the sense of leveraging large-scale auxiliary datasets and more effective due to the capability of optimizing the target task scenarios. More importantly, the proposed model factorization improves the efficacy of model supervised learning and meta-training. Consequently, it achieves eventually more discriminative model adaptation to new tasks. We conducted extensive experiments on two standard benchmark datasets to validate the efficacy of our MBL method in both closed-world and open-world setting, with and without two representative defenses. We also introduced more comprehensive evaluation metrics AUC^2/AUC^m for open-world performance measurement. The results show that the proposed method outperforms the previous state-of-the-art alternatives in the more practical very few-shot learning setting, often by a large margin. **Limitations and future work.** One main limitation with our MBL is that it fails to achieve the best performance when multiple training samples (shots) per website are available. In the future work we will investigate two promising directions. One is to combine meta-learning with conventional supervised learning in order to enjoy their advantages spontaneously. The other is to develop novel data augmentation methods that are particularly suitable for meta-learning methods. We believe both of these studies will significantly advance the progress of few-shot WF attack and make good impact to the community.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2018YFB0204301), and the National Natural Science Foundation of China (No.61472439) under Grant.

References

[1] R. Dingledine, N. Mathewson, P. Syverson, Tor: The second-generation onion router, in: Proceedings of the 13th USENIX Security Symposium, 2004, pp. 303–320.
[2] T. Developers, Tor metrics portal, 2018, (https://metrics.torproject.org).
[3] A. Hintz, Fingerprinting websites using traffic analysis, in: Privacy Enhancing Technologies, 2003, pp. 171–178.
[4] M. Liberatore, B.N. Levine, Inferring the source of encrypted HTTP connections, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, 2006, pp. 255–263.
[5] D.G. Bissias, M. Liberatore, D. Jensen, B.N. Levine, Privacy vulnerabilities in encrypted HTTP streams, in: Privacy Enhancing Technologies, volume 3856, 2006, pp. 1–11.
[6] D. Herrmann, R. Wendolsky, H. Federrath, Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier, in: IEEE International Conference on Cloud Computing Technology and Science, 2009, pp. 31–42.
[7] A. Panchenko, L. Niessen, A. Zinnen, T. Engel, Website fingerprinting in onion routing based anonymization networks, in: Proceedings of the ACM Workshop on Privacy in the Electronic Society, 2011, pp. 103–114.
[8] X. Cai, X.C. Zhang, B. Joshi, R. Johnson, Touching from a distance: website fingerprinting attacks and defenses, in: Preceeding of the ACM Conference on Computer and Communications Security, 2012, pp. 605–616.
[9] T. Wang, I. Goldberg, Improved website fingerprinting on tor, in: Proceedings of the ACM Workshop on Privacy in the Electronic Society, 2013, pp. 201–212.
[10] T. Wang, X. Cai, I. Johnson Roband Goldberg, Effective attacks and provable defenses for website fingerprinting, in: Proceedings of the 23rd USENIX Security Symposium, 2014, pp. 143–157.
[11] J. Hayes, G. Danezis, k-fingerprinting: a robust scalable website fingerprinting technique, in: Proceddings of the 25th USEUIX Security Symposium, 2016, pp. 1187–1203.
[12] A. Panchenko, F. Lanze, M. Henze, Website fingerprinting at internet scale, in: Proceedings of the 16th Network and Distributed System Security Symposium, 2016.
[13] Y. LeCun, Y. Bengio, G.E. Hinton, Deep learning, Nature 521 (2015) 436–444.
[14] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, W. Joosen, Automated website fingerprinting through deep learning, in: Proceedings of the Network and Distributed System Security Symposium, 2018.
[15] P. Sirinam, M. Imani, M. Juarez, M. Wright, Deep fingerprinting: Undermining website fingerprinting defenses with deep learning, in: Proceedings of the ACM Conference on Computer and Communications Security, 2018, pp. 1928–1943.

[16] S. Bhat, D. Lu, A. Kwon, S. Devadas, Var-CNN: a data-efficient website fingerprinting attack based on deep learning, Proceedings on Privacy Enhancing Technologies 2019 (4) (2019) 292–310.
[17] P. Sirinam, N. Mathews, M.S. Rahman, M. Wright, Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning, in: Proceedings of the ACM Conference on Computer and Communications Security, 2019, pp. 1131–1148.
[18] M. Chen, Y. Wang, Z. Qin, X. Zhu, Few-shot website fingerprinting attack with data augmentation, Security and Communication Networks 2021 (2021).
[19] M. Chen, Y. Wang, H. Xu, X. Zhu, Few-shot website fingerprinting attack, Comput. Networks 198 (2021) 108298.
[20] K.P. Dyer, S.E. Coull, T. Ristenpart, T. Shrimpton, Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail, in: Proceedings of the 33rd Annual IEEE Symposium on Security and Privacy, 2012, pp. 332–346.
[21] X. Cai, R. Nithyanand, T. Wang, R. Johnson, I. Goldberg, A systematic approach to developing and evaluating website fingerprinting defenses, in: Proceedings of the ACM Conference on Computer and Communications Security, 2014, p. 227238.
[22] T. Wang, I. Goldberg, Walkie-talkie: An effective and efcient defense against website fingerprinting, in: Proceeding of the 26th USENIX Security Symposium, 2017, pp. 1375–1390.
[23] M. Juarez, M. Imani, M. Perry, C. Diaz, M. Wright, Toward an efficient website fingerprinting defense, in: Proceeding of the European Symposium on Research in Computer Security, volume 9878, 2016, pp. 27–46.
[24] J. Gong, T. Wang, Zero-delay lightweight defenses against website fingerprinting, in: 29th USENIX Security Symposium (USENIX Security 20), USENIX Association, 2020, pp. 717–734.
[25] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, A. Panchenko, Trafficsliver: Fighting website fingerprinting attacks with traffic splitting, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '20, 2020, p. 19711985. New York, NY, USA
[26] T. Wang, I. Goldberg, On realistically attacking tor with website fingerprinting, in: Proceedings on Privacy Enhancing Technologies (PoPETs), 2016, pp. 21–36.
[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P.P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (2011) 2493–2537.
[28] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks (2012) 1097–1105.
[29] E.P. Ijjina, C.K. Mohan, Human action recognition in RGB-D videos using motion sequence information and deep learning, Pattern Recognit. 72 (2017) 504–516.
[30] E. Corona, G. Alenyà, A. Gabas, C. Torras, Active garment recognition and target grasping point detection using deep learning, Pattern Recognit. 74 (2018) 629–641.
[31] W. Xie, X. Jia, L. Shen, M. Yang, Sparse deep feature learning for facial expression recognition, Pattern Recognit. 96 (2019).
[32] A.M. Atto, A. Benoît, P. Lambert, Timed-image based deep learning for action recognition in video sequences, Pattern Recognit. 104 (2020) 107353.
[33] X. Bai, X. Wang, X. Liu, Q. Liu, J. Song, N. Sebe, B. Kim, Explainable deep learning for efficient and robust pattern recognition: a survey of recent developments, Pattern Recognit 120 (2021) 108102.
[34] I. Lauriola, C. Gallicchio, F. Aiolli, Enhancing deep neural networks via multiple kernel learning, Pattern Recognit 101 (2020) 107194.
[35] A. Sanakoyeu, M.A. Bautista, B. Ommer, Deep unsupervised learning of visual similarities, Pattern Recognit 78 (2018) 331–343.
[36] L. Minh Dang, K. Min, H. Wang, M. Jalil Piran, C. Hee Lee, H. Moon, Sensor-based and vision-based human activity recognition: a comprehensive survey, Pattern Recognit 108 (2020) 107561.
[37] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014).
[38] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[39] A.V. Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: a generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).
[40] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions (2016).
[41] S. Wang, L. Wang, S. Yin, H. Zhao, H. Shentu, CPWF: Cross-platform website fingerprinting based on multi-similarity loss, in: 2020 International Conference on Networking and Network Applications (NaNA), IEEE, 2020, pp. 73–80.
[42] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: international conference on machine learning, 1126–1135.
[43] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: a survey, arXiv:2004.05439 [cs, stat] (2020).
[44] Q. Sun, Y. Liu, T.-S. Chua, B. Schiele, Meta-transfer learning for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 403–412.
[45] J.-M. Perez-Rua, X. Zhu, T.M. Hospedales, T. Xiang, Incremental few-shot object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13846–13855.
[46] S. Christian, W. Liu, Y.Q. Jia, Going deeper with convolutions, IEEE Conference on Computer Vision and Pattern Recognition (2015) 1–9.
[47] S. Xie, R. Girshick, P. Dollar, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2017, pp. 5987–5995, doi:10.1109/CVPR.2017.634.
[48] A.F. Agarap, Deep learning using rectified linear units (reLU), arXiv preprint arXiv:1803.08375 (2018).
[49] M. Juarez, S. Afroz, G. Acar, C. Diaz, R. Greenstadt, A critical evaluation of website fingerprinting attacks, Computer and Communications Security, 2014.
[50] Z. Cheng, X. Zhu, S. Gong, Face re-identification challenge: are face recognition models good enough? Pattern Recognit 107 (2020) 107422.
[51] M. Juarez, M. Imani, M. Perry, C. Diaz, M. Wright, Toward an efficient website fingerprinting defense, volume 9878, 2016, pp. 27–46.

Mantun Chen received the bachelor's and master's degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, in 2006 and 2014. He is currently pursuing a Ph.D. with the School of Computer Science, National University of Defense Technology. His research interests include network security and artificial intelligence.

Yongjun Wang received the Ph.D. degree in computer architecture from the National University of Defense Technology, China, in 1998. He is currently a Professor with the College of Computer, National University of Defense Technology. His research interests include network security and system security.

Xiatian Zhu is a Senior Lecturer with Surrey Institute for People-Centred Artificial Intelligence, and the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, UK. He received his Ph.D. degree from the Queen Mary University of London. He won the Sullivan Doctoral Thesis Prize 2016. He was a research scientist at Samsung AI Centre, Cambridge, UK. His research interests include computer vision and machine learning.