Efficient Fine-Grained Website Fingerprinting via Encrypted Traffic Analysis with Deep Learning

Meng Shen*, Zhenbo Gao†, Liehuang Zhu*, Ke Xu‡
*School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, P. R. China
†School of Computer Science, Beijing Institute of Technology, Beijing, P. R. China
‡Tsinghua University & BNRist, Beijing, P. R. China
{shenmeng, gaozhenbo07, liehuangz}@bit.edu.cn; xuke@tsinghua.edu.cn

***Abstract*—*Fine-grained* website fingerprinting (WF) enables potential attackers to infer individual webpages on a monitored website that victims are visiting, by analyzing the resulting traffic protected by security protocols such as TLS. Most existing studies focus on WF at the granularity of website, which takes website homepages as their representatives for fingerprinting. Fine-grained WF can reveal more user privacy, such as online purchasing habits and video-viewing interests, and can also be employed for web censorship. Due to striking similarly of webpages on a same website, it is still an open problem to conduct fine-grained WF in an accurate and time-efficient way.**

In this paper, we propose BurNet, a fine-grained WF method using Convolutional Neural Networks (CNNs). To extract differences of similar webpages, we propose a new concept named unidirectional burst, which is a sequence of packets corresponding to a piece of HTTP message. BurNet takes as input unidirectional burst sequences, instead of bidirectional packet sequences, which makes it applicable to local and remote attack scenarios. BurNet employs CNNs to build a powerful classifier, where sophisticated architecture is designed to improve classification accuracy while reducing time complexity in training. We collect real-world datasets from two well-known websites and conduct extensive experiments to evaluate the performance of BurNet. The closed-world evaluation results show that BurNet outperforms the state-of-the-art methods in both attack scenarios. In the more realistic open-world setting, BurNet can achieve 0.99 precision and 0.99 recall. BurNet is also superior to its CNN-based counterparts in terms of training efficiency.

***Index Terms*—Fine-grained website fingerprinting, encrypted traffic, unidirectional burst, CNNs**

I. Introduction

With the widely adoption of Transport Layer Security (TLS) protocol on websites, the proportion of web traffic encrypted by security protocols keeps on growing [1]. Google has protected 95% of its products and services to ensure security requirements [2]. Website fingerprinting (WF) via encrypted traffic analysis enables potential attackers to recognize the websites visited by victims. Existing studies usually consider the granularity of website, which takes the homepages of websites as their representatives. However, we argue that *fine-grained* WF, which identifies specific webpages of a designated website, can reveal more users privacy, such as the products they viewed on online shopping malls, or videos on YouTube they watched. Fine-grained WF can also be employed by network administrators for web censorship, e.g., blocking access to certain pages on a well-known website.

978-0-7381-3207-5/21/$31.00 © 2021 IEEE

TABLE I: Comparison of existing WF methods
(H: High; M: Medium; L: Low)

	[4]	[5]	[3]	[6]	[7]	BurNet
Accuracy	L	L	M	M	M	H
Applicability to different scenarios	Y	Y	N	N	N	Y
Practicality	N	Y	Y	Y	Y	Y
Automatically learning features	N	N	N	N	Y	Y

Designing an *efficient* fine-grained WF method is extremely challenging. *First*, it is significantly difficult to *accurately* identify encrypted network flows for different webpages, as webpages on the same website have more similarities compared to index pages of different websites [3]. *Second*, it is non-trivial to make a WF method applicable to different attack scenarios. Potential attackers can be located at different positions along the paths from victims to web servers, leading to discrepancy in capability of obtaining network traffic. The *third* challenge lies in *practicality*, meaning that a WF method should work with different versions of protocols, ranging from HTTP to TLS, and achieves time efficiency in training for practical usage.

Existing WF solutions either leverage hand-crafted features for training traditional machine learning classifiers, such as SVM [3] and Random Forest [6], or apply deep learning techniques such as convolutional neural networks (CNNs) on original packet sequence [7]. These methods mostly focus on constructing fingerprinting for index pages of websites and thereby fail to achieve high accuracy in distinguishing similar webpages (c.f. Section VI). In addition, they commonly assume that both uplink and downlink traffic of victims can be obtained by attackers and thus cannot be easily generalized to different attack scenarios.

In this paper, we propose BurNet, an accurate and efficient CNNs-based classifier for fine-grained WF. To abstract discrepant capabilities of attackers, we consider two typical scenarios, where *local* attackers can obtain bidirectional traffic of victims, as commonly assumed in existing studies, while remote attackers can only acquire victims' unidirectional traffic. BurNet is restricted to use only unidirectional traffic for

fingerprinting, making it applicable in both scenarios.

The design of BurNet is motivated by the fact that the differences of encrypted traffic among different webpages are inherently determined by discrepancy of webpage contents, such as the number of elements as well as their types and transmission order. Due to the similarity of webpages on the same website, packet-level statistics or timing information, which is commonly used in existing studies, cannot effectively capture such discrepancy. Therefore, we propose the concept of unidirectional burst, which is a sequence of packets corresponding to the transmission of a piece of application-layer message. Unidirectional burst is a higher-level abstraction over the original packets and can successfully capture the differences of webpages. Using downlink burst sequence as input, we construct an effective classifier based on CNNs, which does not require manually selecting features.

The differences of BurNet from previous WF methods are summarized in Table I. The main contributions of this paper are as follows:

1) By analyzing data encapsulation of HTTP, TLS and TCP, we make an observation that unidirectional burst can reflect webpage differences. We design an efficient method to calculate the sequence of unidirectional burst lengths, which is adaptive to different TLS and HTTP versions.
2) We propose BurNet, which is a CNNs-based classifier with sophisticated architecture design. Compared with existing CNN-based classifiers, BurNet greatly reduces the number of parameters in neural networks and significantly improves training efficiency. It also learns time invariant features from downlink burst sequences, which helps achieve high accuracy.
3) We evaluate the effectiveness of BurNet with the closed-world setting. In the local attacker scenario, BurNet reaches more than 96% accuracy on two real-world datasets, outperforming the state-of-the-art. In the remote attacker scenario, BurNet maintains high accuracy while its counterparts degrade significantly.
4) In a more realistic open-world setting, BurNet can achieve 0.99 precision and 0.99 recall in binary classification, which significantly outperforms its counterparts.

The rest of the paper is organized as follows. We first review the related work in Section II, and then describe the threat model in Section III. We elaborate on the motivation of our work in Section IV and present the design of BurNet in Section V. We evaluate the proposed method and make a comprehensive comparison with existing methods in Section VI, and conclude this paper in Section VII.

II. Related Work

Encrypted traffic analysis has attracted intensive attention, including malicious traffic detection [8], APP traffic classification [9, 10], video traffic classification [11, 12], and website fingerprinting [13, 14]. From the perspective of the method used, the related work can be divided into two categories. One is to use artificially designed features and traditional machine learning algorithm as classifier. The other is to use the technology of deep learning to train classifiers. We briefly summarize the existing methods in Table II.

TABLE II: Summary of existing methods

Categories	Models	Features
Traditional machine learning	MC [4, 15]	SSL/TLS message type
	k-NN [5]	Uplink time information
	SVM [3]	Cumulative packet length
	k-NN [6]	Statistical features
	RF [16]	Packet length information
Deep learning	SDAE [19]	Sequence of packet direction
	CNN [7, 13]	Sequence of packet direction
	FlowPic [20]	Packet size-arrival time histogram
	BurNet (**This paper**)	Sequence of unidirectional burst lengths

A. Traditional Machine Learning Methods

Several studies employ traditional machine learning models to build classifiers. SSL/TLS message type is used to construct Markov Chain (MC) [4, 15]. Shen et al. [4] use second-order Markov chain and application attribute bigrams to classify encrypted traffic. However, TLS 1.3 only needs 1-RTT to complete handshake, which reduces the available SSL/TLS message types. The methods based on packet length usually give different attributes to packet length in different directions [3, 16]. CUMUL [3] sets positive and negative attributes for packet length in different directions, and uses cumulative packet length as features to train an SVM classifier, which has a good performance in detection accuracy. Statistical features such as maximum, minimum, and average length, are also used for fingerprinting [6, 9, 17]. k-fingerprints (k-FP) [6] uses random forest (RF) to extract fingerprinting vector from the statistical features. The obtained fingerprinting vector is then used by k-Nearest Neighbor (k-NN) algorithm to build classifiers. However, these methods need design and select features [18], and these features are not discriminative enough for fine-grained WF.

B. Method Based on Deep Learning

Deep learning (DL) has been employed in WF, which can automatically learn high-level features from input. These methods [7, 13, 19] usually use -1 and 1 to represent packet directions. Deep Fingerprinting (DF) [7] leverages a CNN with many convolutional layers, which can learn features from simple direction sequence for WF. However, the direction sequence is no longer distinguishable when there is only unidirectional traffic. Shapira et al. [20] also propose a CNN-based classifier named FlowPic, which transforms each traffic flow into an image and turns WF into image classification. It requires heavy resource overhead because the transformed image is much larger than the input of other fingerprinting methods (c.f. Section VI-B).

In this paper, we propose the concept of unidirectional burst, which build the relationship between HTTP messages and their corresponding packets. The unidirectional burst sequence is

used for WF, and thus space complexity is greatly reduced compared with the original packet sequence. We use CNNs without fully-connected layers to learn high-level features from unidirectional burst sequences, which can achieve training efficiency and classification accuracy simultaneously.

III. Problem Description

This section describes the threat models for launching fine-grained website fingerprinting attacks by potential adversaries, and also illustrates the closed- and open-world settings that are commonly used to assess the effectiveness of specific fingerprinting methods.

A. Threat Model

Fine-grained website fingerprinting aims at inferring which webpages in a website are visited by a victim, through carefully analyzing the encrypted traffic generated by the victim's web browsing behaviors. Following the common assumption in the literature [7], we assume a *passive* attacker, which means that the attacker can only eavesdrop on encrypted packets through network devices, but cannot drop, modify, or decrypt network packets. The fingerprinting is conducted at the granularity of *flows*, where a flow is uniquely defined by the traditional five-tuple, namely source IP/port, destination IP/port, and transport protocol. In practice, an attacker first recognizes the flows belonging to the target website through the Server Name Indication (SNI) in the *Client Hello* packet of the TLS protocol and then uses the recognized flows to perform the webpage-level fingerprinting.

The victim accesses the Internet to fetch webpage contents via browsers, as illustrated in Fig. 1. We focus on two types of threat models, where the potential attackers are located in different positions, resulting in different capabilities of collecting network packets.

Local attackers. The attackers are located in the *local* network of the victim, where they can eavesdrop on the *bidirectional* traffic between the victim and the corresponding web servers. Such attackers can be campus network administrators or local Internet Service Providers (ISPs). They can obtain both uplink and downlink traffic traversing through the gateway of local networks.

Remote attackers. The attackers are located far from the victim, e.g., on a certain critical point along the paths between the victim and the corresponding web servers. Such attackers may be located at critical routers in large autonomous systems (ASes) or regional ISPs, leveraging fingerprinting results for web censorship, e.g., banning users in a region from accessing certain webpages in a website. Since asymmetric routing (e.g., hot-potato routing strategy [21]) commonly exist in the Internet, i.e., the uplink and downlink traffic of the victim usually follows different routes. Thus, only *unidirectional* traffic of the victim is visible to the remote attackers.

Note that even if a remote attacker is powerful enough to control all area boarder routers, e.g., a regional ISP, recovering bidirectional traffic of the victim is prohibitively expensive, because heavy time and space overhead is required to merge and reorder packet sequences collected from different routers.

The remote attacker model imposes more constrains on the capability of attackers, and thus is more challenging to design effective fingerprinting methods. Different from most existing studies that consider merely the local attacker model, we consider the two models as a whole in this paper.

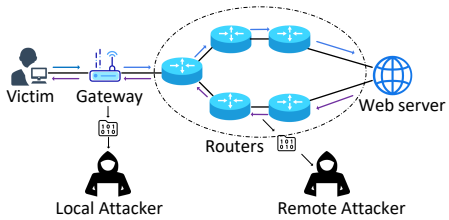


Fig. 1: Two typical attack scenarios for fine-grained WF

B. Assessment Scenarios

Closed-world setting. It assumes that the victim can only visit a fixed collection of *monitored* webpages from the same website. It is used to evaluate the ability of a fingerprinting method to distinguish similar webpages.

Open-world setting. It assumes that the victim not only visits a collection of monitored webpages, but also visits a larger number of *unmonitored* webpages. It is used to evaluate a fingerprinting method in a more *realistic* scenario, which can avoid the base rate fallacy [3]. In such a scenario, the attackers can first conduct binary classification to recognize the monitored webpages from unmonitored ones, and then perform multi-classification to identify which monitored webpage it is.

IV. Motivation

Webpage fingerprinting relies on discovering critical differences of webpages in terms of the resulting encrypted traffic. Existing fingerprinting methods attempt to capture such differences by using statistics, timing, TLS state transition, or packet length features. These features, however, are not discriminative enough for distinguishing webpages on the same website, as demonstrated in Section VI.

Our work is motivated by the fact that the differences of encrypted traffic are inherently determined by the divergence of webpages, including the types of elements (e.g., text, image and video) as well as their sizes and loading orders. Thus, we will have a deep look at the webpage fetching process and try to *link* element loading with the resulting encrypted packet sequences, which provides us with a fresh angle to discover and abstract discriminative features.

A. From Webpage Elements to Encrypted Packets

We take as an example the typical webpage fetching process to analyze the generation of encrypted packets from webpage elements. For visiting a certain webpage, a user enters the corresponding URL in a browser, which establishes a connection between the client and the web server. In order to fetching each webpage element, the client sends a HTTP request to the

server, who, in turns, replies with a HTTP response that contains the required element. Each element usually corresponds to one or multiple HTTP messages. Transport Layer Security (TLS) protocol has been widely used to secure the client-server communication.

For ease of illustration, we visualize the data encapsulation process in Fig. 2, which exhibits the relationship between a message and the resulting encrypted packets. TLS receives a message from HTTP and divides them into fragments according to record size. Then each fragment is optionally compressed, added with a message authentication code (MAC) and encrypted. The fragment plus TLS header forms a TLS record, which becomes the payload of TCP at the transport layer. If the length of TLS record exceeds maximum segment size (MSS), it will be segmented. MSS is set to avoid slowing down the connection speed caused by packet fragmentation at the network layer. Therefore, an IP packet has only one TCP segment.

TLS record size has an important impact on the performance of HTTPS. Here we discuss the effect of record size on data encapsulation, as shown in Figure 2. TCP is a byte-stream protocol, which can split message from the upper layer of TCP (e.g., TLS) in arbitrary ways for transmission. Thus, there is a *length* field in TLS header to make the receiver know where the record ends. For large record size, TLS record exceeding MSS limit will be transmitted in multiple TCP payloads, and only one of these TCP payloads contains the TLS header. While for small record size, the entire TLS record can be accommodated in a single TCP payload. Since the length of TLS record is usually less than MSS, TCP will intercept the next TLS record to pad the current payload as large as MSS. As a result, a TCP payload may contain none, one or multiple TLS headers. On the whole, no matter which record size it is, we can always construct the relationship between packets and their corresponding HTTP messages.

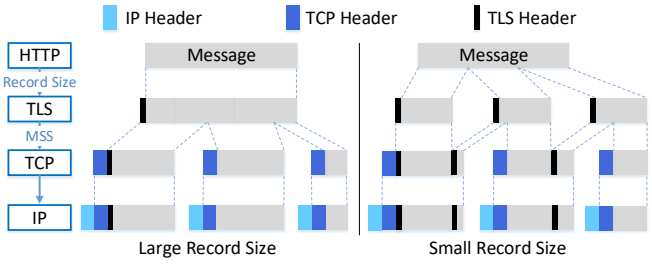


Fig. 2: Data encapsulation of HTTP, TLS, TCP and IP

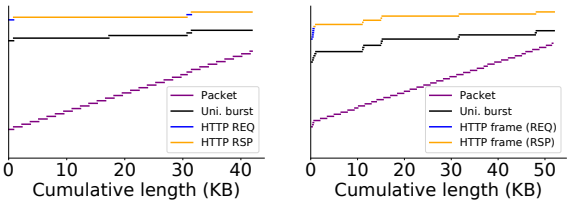
B. Message-Aware Unidirectional Bursts

The analysis above shows that a HTTP message will result in one or multiple successive packets. If we associate the packets in encrypted traffic with the corresponding application-layer message, we can discover the element loading characteristics from encrypted traffic, which helps us extract discriminative features for fingerprinting.

Our idea is motivated by the fact that burst acts a critical role in constructing website fingerprinting [17]. In practice, when bidirectional traffic can be obtained by the attacker, a burst is simply recognized as a subsequence of packets with the same direction. In the unidirectional traffic scenario, however, it is impossible to define burst as before, as all packets are in the same direction.

In this paper, we define the concept of burst from a new perspective, which builds the relationship between packets and application-layer message.

Definition 1. *(Unidirectional burst).* A unidirectional burst is defined as the sequence of unidirectional packets corresponding to each piece of HTTP message.



(a) MicroBlog (HTTP/1.1) (b) Twitter (HTTP/2)

Fig. 3: HTTP messages and their resulting unidirectional bursts and packets

To justify that unidirectional burst can reflect HTTP interactions better than original packets, we select MicroBlog and Twitter as representatives that adopt two mainstream HTTP versions, namely HTTP/1.1 and HTTP/2. The HTTP interactions, as well as the resulting unidirectional bursts and packet sequences are plotted in Fig. 3. The length of each short line denotes the length of an HTTP message, a unidirectional burst, or an encrypted packet. For ease of illustration, all short lines of the same type are shifted vertically.

When visiting a specific webpage of MicroBlog, as shown in Fig. 3(a), an HTTP request is always followed by an HTTP response, because HTTP/1.1 requires that a new HTTP request cannot start until the previous response completes. We can find that the unidirectional bursts, no matter uplink or downlink bursts, are basically consistent with the HTTP messages. Similar trends can also be observed in Fig. 3(b). The major difference is two-fold. First, HTTP/2 adopts smaller-size frames in sending requests and responses, where either a frame corresponds to one or multiple unidirectional bursts, or multiple frames form a single burst. Second, HTTP/2 allows sending all request frames at the beginning, which lead to the arrival of multiple response frames in a certain order.

Now, we have validated that the concept of unidirectional burst can be generalized to different HTTP versions, even though the implementation of these protocols is slightly different among different websites.

Definition 2. *(Unidirectional burst sequence).* Given a unidirectional flow, the unidirectional burst sequence is defined as the sequence of all unidirectional burst lengths in the flow.

In order to make the proposed method applicable in both attack scenarios, we only consider unidirectional traffic for

building webpage fingerprinting. However, it must be admitted that because of more information in bidirectional traffic, if the uplink burst sequence and downlink burst sequence can be extracted from the bidirectional traffic, it is more conducive to the construction of webpage fingerprinting. In this paper, we select the downlink traffic because it is more informative than the uplink traffic. Since the downlink flows are mainly used for carrying HTTP responses, they usually have a larger volume of packets and vary greatly due to the differences in website elements, making it more suitable for constructing fine-grained WF.

Comparing with the original packet length sequence commonly used in recent studies [3], the unidirectional burst sequence is more beneficial to fingerprinting construction:

- The majority of packets in a flow are usually with the maximum length, which reduces the diversity of packet sequences for different webpages. In contrast, the length of unidirectional burst is determined by the length of HTTP message, which potentially captures the differences in terms of website elements.
- The number of unidirectional bursts is far less than that of individual packets, which helps reduce the space and time complexity in training classifiers.

C. Calculation of Unidirectional Burst Sequence

Despite that packets in a unidirectional burst are sent continuously in a short time interval on the server-side, it is impossible for an attacker to identify these bursts through inter-packet time interval, as the inter-packet interval perceived by a local or remote attacker varies significantly due to network fluctuation. Therefore, we propose a method to accurately calculate the unidirectional burst sequence as follows.

Recall that TLS record size leads to differences in data encapsulation (c.f., Fig. 2). For large record size, the length of a unidirectional burst, b, is indicated by the *length* field in TLS header. When small record size is used, the calculation method of unidirectional burst sequence is shown in Algorithm 1. It takes as input a unidirectional flow P. Each packet $p_i \in P$ is associated with three values: $p_i.l$, $p_i.n$, and $p_i.r$, denoting the length of TCP payload, the number of TLS headers in TCP payload, and an array containing the value of the *length* field in TLS header, respectively. Let seg denote the length of the remaining TLS record in the subsequent TCP payloads and b denote the current unidirectional burst. First, seg and b are updated when traversing the TLS headers in the current TCP payload (lines 3-6), and then seg subtracts the length of current TCP payload to get the length of remaining TLS record in the subsequent TCP payloads (line 7). If seg is equal to 0, the current unidirectional burst ends and its length is added into B (lines 8-11). Finally, the unidirectional burst sequence B is obtained.

Note that different TLS versions have the same description of TLS header. i.e., TLS header takes up 5 bytes and contains the *length* field [22]. Therefore, this method can be generalized to different TLS versions.

Algorithm 1 Calculation of unidirectional burst sequence

Input: A unidirectional flow $P = [p_1, p_2, \ldots, p_n]$
Output: $B = [b^{(1)}, b^{(2)}, \ldots, b^{(m)}], m \leq n$;
1: Initialize B as an empty array, $seg = 0$, and $b = 0$;
2: **for** $i = 1$; $i \leq n$; $i++$ **do**
3: **for** $j = 1$; $j \leq p_i.n$; $j++$ **do**
4: $b+ = (p_i.r[j] + 5)$
5: $seg+ = (p_i.r[j] + 5)$
6: **end for**
7: $seg = seg - p_i.l$
8: **if** $seg == 0$ **then**
9: Append b to B
10: $b = 0$
11: **end if**
12: **end for**
13: **return** B;

V. The Proposed BurNet

This section presents the design of BurNet, including system overview, architecture design and function formulations.

A. System Overview

The analysis in the last section shows that downlink burst sequence can capture unique patterns of similar webpages. Thus, we leverage downlink burst sequence to represent a flow and turn the fine-grained WF into the classification problem of downlink burst sequences.

The downlink burst sequence implicitly indicates the order of bursts, which is analogous to time series. Inspired by the recent advances in time series classification (TSC) [23], we employ CNNs to build a powerful classifier, as it can learn space invariant features in input space. CNN has also demonstrated its effectiveness in other fields, such as speech recognition [24] and text classification [25], where audio and text exhibit natural temporal ordering.

The overview of BurNet is shown in Fig. 4. First, encrypted traffic flows are collected from the Internet and then parsed out meta-information to construct downlink burst sequences, as described in Section IV-C. Note that it is efficient to parse the TLS headers in TCP payload, because the position of the next TLS header can be obtained according to the position of the current TLS header plus the current TLS record length, instead of traversing TCP payload by byte to find TLS headers. Then, the CNNs classifier takes each downlink burst sequence B_d as input and learns feature representation. Finally, BurNet predicts the label of the burst sequence, which reveals the webpage it belongs to.

B. CNNs Construction

Although VGG [26] and FCN [27] offer basic network structures, we still need to construct and optimize CNNs according to our requirements.

Problem Formulation. Let D denote the number of similar webpages from the same website. The visits of these webpages result in M downlink burst sequences in total. The i-th sequence is denoted by $B_d^{(i)} = [b_d^{(i,1)}, b_d^{(i,2)}, ..., b_d^{(i,T_i)}]$, where $b_d^{(i,j)}$ is the j-th burst and T_i is the total number of bursts. The

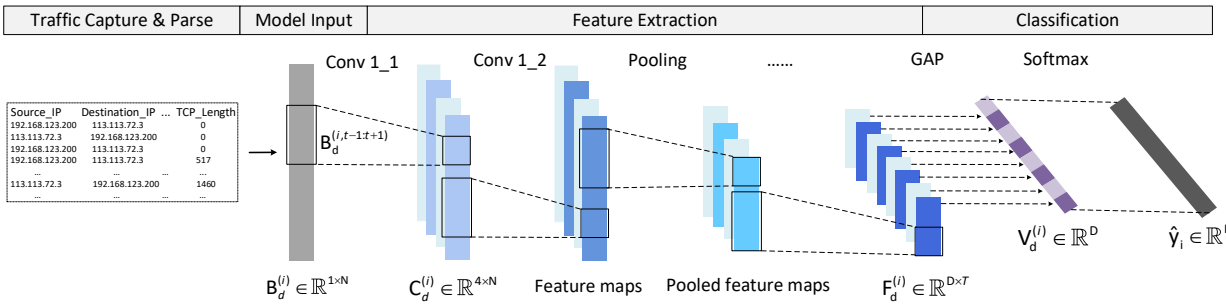


Fig. 4: The system overview of BurNet

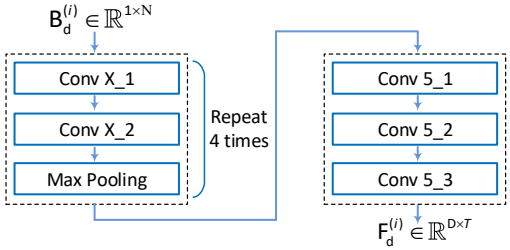


Fig. 5: Network structure for feature extraction in BurNet

real label of $B_d^{(i)}$ is y_i, $1 \leq y_i \leq D$. Taking $B_d^{(i)}$ as input, BurNet makes a prediction according to Eq. (1),

$$\hat{y}_i = \psi(B_d^{(i)}; w), \hat{y}_i \in \mathbb{R}^D \tag{1}$$

where w is the set of BurNet parameters, $\hat{y}_i$ is the probability that the input data $B_d^{(i)}$ is predicted as each class (i.e., label). Our goal is to train $\psi(B_d^{(i)}; w)$ so that the label with the maximum probability in its output is y_i. Note that we use the same architecture in both closed- and open-world settings.

Model Inputs. In order to be able to build webpage fingerprinting quickly, instead of waiting for all the flows to be loaded, we can determine a length threshold $\mathcal{N}$. If a sequence length exceeds the threshold, we intercept the first $\mathcal{N}$ elements from the original input.

Feature Extraction. In our model, there are five blocks for feature extraction, as shown in Fig. 5. In the first four blocks, there are two consecutive convolutional layers before the pooling layer; and in the last block, there are three consecutive convolutional layers without pooling layer. The number of filters in the convolutional layer Conv5_3 is equal to the number of webpage labels (i.e., D). After each convolutional layer, there is a BatchNorm (BN) layer that accelerates the training of neural networks.

The formula for the first convolutional layer Conv1_1 at a centered time step t is given as follows:

$$C_d^{(i)} = ReLU(\mathcal{W}_\mathcal{B} * B_d^{(i,t-1:t+1)} + b) \mid \forall t \in [1, \mathcal{N}] \tag{2}$$

where $C_d^{(i)}$ denotes the result of a convolution applied on B_d^i of length $\mathcal{N}$ with a filter $\mathcal{W}_\mathcal{B}$ of length 3, a bias parameter b and a non-linear function $ReLU$. The formula for other convolutional layers is similar to Eq. (2). The activation function $ReLU$ increases the nonlinearity of neural networks and improves the expression ability of neural networks. The weight sharing property of the convolution layer enables them to learn filters that are invariant across time dimension.

Let $F_d^{(i)} \in \mathbb{R}^{D \times T}$ be the output of Conv5_3, that is, the feature representations of the input data $B_d^{(i)}$. D is the number of channels and is equal to the number of webpage labels, and T is the length of the feature map.

Classification. BurNet replaces traditional fully-connected layer with a Global Average Pooling (GAP) layer, which significantly reduces the amount of parameters, thus greatly improving training speed and reducing the risk of over fitting. The GAP layer accepts the output of Conv5_3 and averages each feature map in time dimension, as shown in Eq. (3),

$$V_d^{(i)} = \frac{1}{T} \sum_{j}^{T} F_d^{(i,\beta,j)} \mid \forall \beta \in [1, D] \tag{3}$$

where $V_d^{(i)} \in \mathbb{R}^D$ is the output of GAP. Finally, a traditional softmax classifier is applied to the output, as shown in Eq. (4),

$$\hat{y}_i = softmax(\mathcal{W}_\mathcal{V} * V_d^{(i)}) \tag{4}$$

where $\mathcal{W}_\mathcal{V}$ are the parameters.

Loss Function. WF is essentially a multi-classification task, so we use Cross-Entropy loss function to calculate the loss between the predicted values $\hat{y}$ and the ground truth y,

$$L = -\frac{1}{|X|} \sum_{i}^{|X|} \sum_{j}^{D} I(y_i = j) log(\hat{y}_{ij}) \tag{5}$$

where $|X|$ is the number of model inputs X and $I(y_i = j) = 1$ if y_i is j, else 0. We use the Adam optimizer to minimize the loss during training.

Avoid Over-fitting. There is a risk of over-fitting in the training process of CNNs. In BurNet, we use the following measures to prevent over-fitting. First, we add a dropout layer at the end of each block. In the forward propagation, the dropout layer will make the neurons stop working with a certain probability, so that the model will not rely on some local features too much, thus improving the generalization ability. Second, in order to reduce the complexity of the model, we introduce L2 regularization. That is adding the sum of

squares of weight parameters on the basis of the original loss function. The new loss function $\mathcal{L}$ is defined in Eq. (6),

$$\mathcal{L} = L + \lambda \sum_{k} w_k^2 \tag{6}$$

where λ is a hyperparameter that controls the size of the regular term. The L2 regularization tends to make all parameters w smaller to constrain the complexity of the model.

Learning Rate. Learning rate is an important hyperparameter of neural network optimization. In BurNet, exponential learning rate decay is used. In the early stage, a larger learning rate is used to accelerate convergence, and in the later stage, a smaller learning rate is used to ensure stability. The formula of learning rate decay is shown in Eq. (7),

$$\alpha_k = \alpha_0 . \alpha_1^{k\mathcal{E}^{-1}} \tag{7}$$

where α_k is the learning rate at the k-th epcho, α_0 is the initial learning rate, $\alpha_0.\alpha_1$ is the learning rate at the end of training, and $\mathcal{E}$ is the number of epochs.

VI. Performance Evaluation

In this section, we evaluate the effectiveness of BurNet with real-world datasets collected from two well-known websites. We first introduce the experimental setup, then tune the BurNet's hyperparameters, and finally make a comprehensive evaluation of BurNet with the state-of-the-art.

A. Experimental Setup

1) Methods to Compare: Our preliminary experiments find that MC-based method [4] and DTW [5] result in low accuracy (less than 30%), so we do not consider them for comparison. We select four representative WF methods described as follows. Note that we have to make modifications on some methods for comparison in remote attack scenario, as they are originally designed only for local attack scenario. All fingerprinting methods are trained on a server equipped with an Inter Core Duo 3.6GHz, 32GB of memory, and a GPU with 8GB of memory.

- *CUMUL* [3] is a WF method using cumulative packet length. CUMUL sets positive or negative signs for packet length in different directions. In remote attack scenario, all packet lengths are positive or negative.
- *k-FP* [6] is a method using statistical features for fingerprinting. In the remote attack scenario, the features of uplink traffic will be removed.
- *DF* [7] is a CNNs-based method that uses the sequence of packet direction as its input. Only one direction is left in the remote attack scenario.
- *FlowPic* [20] transforms the flows of each webpage into an image and uses a CNNs-based classifier for prediction.

2) Dataset Construction: In order to verify the generalizability of BurNet, we build datasets by visiting similar webpages on two famous websites, namely JD [28] and YouTube [29]. The standard of similarity lies in that webpages have the same or similar layout.

We select a group of webpages as the *monitored* webpages, each of which is visited dozens or even hundreds of times. We also choose another group of *unmonitored* webpages that are hundreds of times larger in volume than the previous group, where each unmonitored webpage is visited only once.

We use Selenium to drive Google Chrome to access webpages and use tcpdump to capture the traffic generated by visiting the webpages. To emulate user visits from different locations, the data collection is conducted on Ali cloud servers located in Virginia, Silicon Valley, Japan, Qingdao, Beijing and Chengdu. Our dataset collection took three months. We use the SNI field to refine the traffic flows for each dataset, e.g., "yt3.ggpht.com for YouTube and "imgXX.360buying.com" for JD.

The datasets obtained after the flows are parsed are shown in Table III. For the closed-word setting, YouTube (YT) dataset contains a total of 29,902 sequences from 103 webpages. The JD dataset contains 12,155 sequences from 78 webpages. For the open-word setting, excluding those in the closed-world datasets, we visit 10,000 YouTube webpages and 6,500 JD webpages, respectively. After removing data of failed visits, the open-world dataset contains 9,767 YouTube sequences and 6,477 JD sequences.

TABLE III: Dataset overview

	Closed-world			Open-world	
	Monitored	Sequences	Labels	Unmonitored	Sequences
YT	103	29,902	103	9,767	9,767
JD	78	12,155	78	6,477	6,477

3) Cross Validation: We use 10-fold cross validation for evaluation. That is, we divide the dataset into 10 parts, and take turns to train classifiers on 9 parts and test with the rest one. The mean value of the results of 10 times is used as the result of each method.

B. BurNet's Hyperparameter Tuning

We use pytorch to implement the deep learning model of BurNet. In order to get a model with strong generalization ability, we need to adjust the hyperparameters in the model. We determine a hyperparameter space based on experience, and then find the optimal values in the hyperparameter space, as exhibited in Table IV. When training with different datasets, we only need to tune three hyperparameters, i.e., $\mathcal{N}$, α_0 and λ, greatly reducing time spent on tuning hyperparameters.

Figure 6 shows the impact of the number of epochs on accuracy and error rate of BurNet on YT dataset. Similar results can be obtained on JD dataset, which is omitted here. Its accuracy on the testing set exceeds 96% after only 10 epochs and grows with the increase of epochs. More importantly, the differences between training accuracy and testing accuracy are always less than 2%, indicating that BurNet can avoiding overfitting.

In our preliminary experiments, we find that FlowPic [20] requires extremely large time and space costs in training. To make a fair comparison, we compare the space and time

TABLE IV: Hyperparameter selection of BurNet

Hyperparameter	Search Range	Final
$\mathcal{N}$	[100...500]	300
Optimizer	[Adam,Adamax,SGD]	Adam
α_0	[0.001...0.01]	0.005
α_1	[0.1...0.5]	0.2
EPOCH	[20...100]	30
Batch Size	[50...200]	150
λ	[0.0001...0.1]	0.001
Activation Functions	[Tanh,ReLU,Sigmoid]	ReLU
Number of Filters		
Block1[Conv1_1,Conv1_2]	[2...8]	[4,4]
Block2[Conv2_1,Conv2_2]	[4...16]	[8,8]
Block3[Conv3_1,Conv3_2]	[8...32]	[16,16]
Block4[Conv4_1,Conv4_2]	[16...64]	[32,32]
Block5[Conv5_1,Conv5_2]	[32...128]	[64,64]

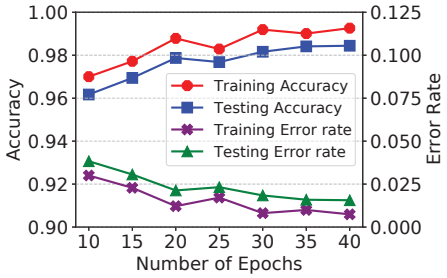


Fig. 6: Closed-world: Impact of the number of epochs on BurNet's accuracy and error rate on YT dataset

overhead required by FlowPic and BurNet to achieve 90% training accuracy on two datasets in closed-world setting, as summarized in Table V.

Each input of BurNet occupies about 300 floats, which is around 1.17K if each float takes up 4 Bytes. However, each input of FlowPic takes up 1500×1500 floats, which is 7500 times more than that of BurNet. Thus, FlowPic results in a large volume of input data for training, which cannot be completely loaded in memory. In our implementation, FlowPic employs eight threads to load data. Its time consumption is significantly larger than BurNet. FlowPic is not an efficient and resource-friendly WF method, thereby we no longer consider it in further comparisons.

TABLE V: Comparison of overhead of FlowPic and BurNet

	Space (KB)	Training Time (s)	
		YT	JD
FlowPic	8,789.06	7,809.93	4,038.16
BurNet	1.17	21.92	14.11

C. Closed-World Evaluation

Now we evaluate the accuracy of each method in closed-world setting. The *accuracy* of each method is defined as the proportion of all instances that are classified correctly.

Table VI shows the accuracy of different methods on two datasets in two attack scenarios. In local attack scenario, BurNet uses downlink traffic and the other methods use bidirectional traffic (The reason why BurNet uses downlink traffic is shown in Section IV-B); while in remote attack scenario, all methods leverage only downlink traffic.

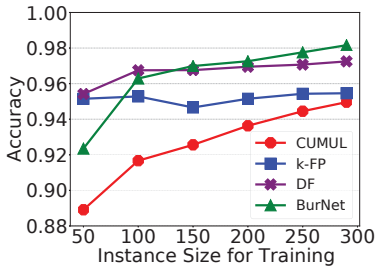


Fig. 7: Closed-world: Impact of training data size of YT dataset on accuracy in local attack scenario

We make two key observations from the results.

1) In local attack scenario, BurNet achieves the highest accuracy on two datasets, although the other methods uses bidirectional traffic for training. The results demonstrate that the downlink burst sequence as well as the CNNs classifier proposed in this paper can extract discriminative features in similar webpages.

2) In the remote attack scenario, BurNet can work well, whereas the accuracy of other methods drops dramatically compared with the results in local cases. In particular, DF exhibits the most significant decrease in accuracy when only unidirectional traffic is available. The results also confirm that the remote attack scenario imposes more challenges on WF methods, as unidirectional traffic leaks less information.

TABLE VI: Closed-world: Accuracy of different methods on two datasets in two scenarios

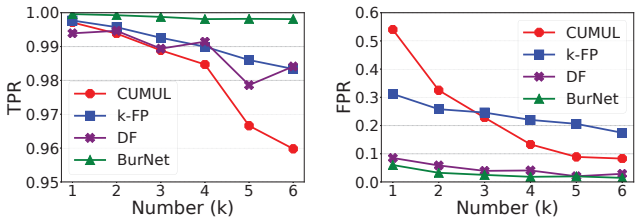
	Local Attack Scenario				Remote Attack Scenario			
	CUMUL	k-FP	DF	BurNet	CUMUL	k-FP	DF	BurNet
YT	0.949	0.954	0.972	0.981	0.747	0.822	0.068	0.981
JD	0.499	0.901	0.808	0.966	0.168	0.824	0.045	0.966

Figure 7 shows the impact of the size of YT dataset on accuracy. The experiment is carried out in local attack scenario and the number of instances of each label for training varies from 50 to 290. The results show that the accuracy of each method grows with the increase of training instances, especially for CUMUL and BurNet. When the number of instances of each label exceeds 150, BurNet outperforms the rest methods and can further improve its accuracy as the training data size increases.

D. Open-World Evaluation

We now evaluate the methods in a more realistic open-world setting, where the attacker needs to determine whether the traffic comes from a monitored webpage or an unmonitored one. Motivated by the experiments in the literature [3, 7, 13], we focus on the *binary* classification, that is, whether an instance can be correctly classified as monitored or unmonitored.

We use prediction probability to label instances. If an monitored instance has a prediction probability greater than

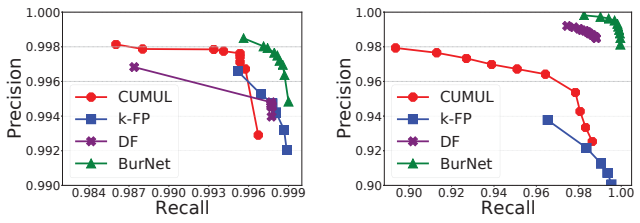


(a) TPR (b) FPR

Fig. 8: Open-world: TPR and FPR with varying amounts of unmonitored webpages on JD dataset in local attack scenario

a certain threshold, it is considered as a true positive (TP), otherwise a false negative (FN). Similarly, correctly labeling an unmonitored instance leads to a true negative (TN), otherwise a false positive (FP). Several metrics are used for evaluation [3, 6, 7]. True positive rate (TPR) is $TP/(TP+FN)$ and false positive rate (FPR) is $FP/(FP+TN)$. Precision and recall are also used to avoid the base rate fallacy, which are defined as $TP/(TP+FP)$ and $TP/(TP+FN)$, respectively. In practice, the attacker can adjust the threshold to make trade-offs between precision and recall.

Figure 8 shows TPR and FPR with varying number of unmonitored webpages on JD dataset. Recall that JD dataset has 12,155 sequences from 78 monitored webpages. We change the number of unmonitored webpages from 1,000 to 6,000. The results show that both TPR and FPR of each method tend to decline with the increase of unmonitored webpages. However, the number of unmonitored webpages has a greater impact on CUMUL and k-FP, while BurNet is less affected. BurNet and DF have the similar FPR trend as the number of unmonitored webpages increases, but TPR of BurNet is higher than that of DF. The downward trend of FPR tends to flatten after the number of unmonitored webpages exceeds 5,000.



(a) YT dataset (b) JD dataset

Fig. 9: Open-world: Precision-Recall curves in local attack scenario

Next, we fix the number of unmonitored webpages in the dataset and use precision and recall to explore the performance of different methods in both local and remote attack scenarios. The YT dataset consists of 29,902 sequences from 103 monitored webpages and 9,677 sequences from 9766 unmonitored webpages. The JD dataset includes 12,155 sequences from 78 monitored webpages and 3,819 sequences from 3,819 unmonitored webpages. The ratio of monitored to unmonitored sequences in both datasets is about 3:1.

Figure 9 shows the precision-recall curves of YT and JD datasets in local attack scenario. Precision-recall curve is an important indicator to evaluate the performance of classifier in unbalanced datasets. By adjusting the prediction threshold, the precision and recall of all methods on YT dataset can reach 0.994 at the same time, which is better than that on JD dataset.

BurNet and DF are highly efficient on YT and JD datasets, and BurNet performs even better than DF. We notice that the precision of k-FP on the two datasets is lower than other methods. The possible reason is that the similarity of webpages leads to less discriminative statistical features used by k-FP.

TABLE VII: Open-world: Precision and recall in remote attack scenario

	YT		JD	
	Precision	Recall	Precision	Recall
CUMUL	0.9837	0.9904	0.8217	0.9688
k-FP	0.9773	0.9837	0.9014	0.9627
DF	0.7537	1.0000	0.7609	0.9999
BurNet	0.9969	0.9988	0.9910	0.9975

Table VII shows precision and recall of different attacks on two datasets in the remote attack scenario. The threshold of true positive is set as 0.5. BurNet can achieve 0.99 precision and 0.99 recall simultaneously on both datasets, which is better than other methods. Comparing with the results in Fig. 9, the precision of other methods in remote attack scenario degrade greatly, especially on JD dataset. For instance, DF tends to predict all instances as monitored, so its recall on both datasets approaches 1.0 whereas the precision is relatively low.

E. Training Cost

TABLE VIII: Closed-world: Training time on YT and JD datasets

	CUMUL	k-FP	DF	BurNet
YT	57.32s	9.73s	182.66s	119.83s
JD	73.12s	3.54s	78.70s	58.33s

We record the training time of various methods in local attack scenario in the closed-world setting. The results in Table VIII show that k-FP takes the least training time. DF and BurNet, which leverage CNNs, take longer time compared with traditional machine learning methods. Although BurNet has deeper convolutional layers, it needs less training time than DF, because it does not include fully-connected layer and thus significantly reduces the amount of parameters for training.

F. Discussion

Through the extensive experiments presented above, we verify the effectiveness of BurNet on fine-grained WF. But there is a limitation in this work. Our two datasets are collected in the general traffic scenario, without considering tunnel traffic scenarios such as Tor. Although the general traffic scenarios account for the vast majority in the real world, the tunnel traffic scenarios are also worth exploring. The unidirectional bursts required by our method can also be obtained from the traffic

of Tor. Therefore, BurNet is potentially applicable to such a scenario and we leave this evaluation as the future work.

VII. Conclusion

In this paper, we presented BurNet, which is an efficient method for fine-grained WF. In order to capture the differences of similar webpages from the perspective of network traffic, we proposed the concept of unidirectional burst. The sequence of unidirectional bursts can make BurNet applicable to both local and remote attack scenarios. We leveraged downlink burst sequences as input and carefully designed a CNNs-based classifier. We collected two real-world traffic datasets and conducted extensive experiments to make a comprehensive comparison of BurNet with the state-of-the-art. The results in both closed- and open-world settings show that, even using only downlink traffic, BurNet outperforms the state-of-the-art methods in terms of accuracy. In future work, we will further evaluate the effectiveness of the proposed method in more attack scenarios.

Acknowledgments

This work is partially supported by National Key R&D Program of China with No. 2020YFB1006101, Beijing Nova Program with No. Z201100006820006, NSFC Projects with No. 61972039, 61932016 and 61872041, Beijing Natural Science Foundation with No. 4192050, Open Research Projects of Zhejiang Lab under Grant 2020AA3AB04, China National Funds for Distinguished Young Scientists with No. 61825204, Beijing Outstanding Young Scientist Program with No. BJJWZYJH01201910003011.

References

[1] NetMarketShare, "Market share for mobile, browsers, operating systems and search engines — netmarketshare," https://netmarketshare.com/, 2019.

[2] google, "Google transparency report," https://transparencyreport.google.com/https/overview?hl=en, 2020.

[3] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[4] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order markov chains and application attribute bigrams," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

[5] S. Feghhi and D. J. Leith, "A web traffic analysis attack using only timing information," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 8, pp. 1747–1759, 2016.

[6] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 1187–1203.

[7] P. Sirinam, M. Imani, M. Juárez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 1928–1943.

[8] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu, "Torward: Discovery, blocking, and traceback of malicious traffic over tor," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 12, pp. 2515–2530, 2015.

[9] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2018.

[10] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 2367–2380, 2021.

[11] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute - encrypted HTTP adaptive video streaming title classification," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 12, pp. 3039–3049, 2017.

[12] M. Shen, J. Zhang, K. Xu, L. Zhu, J. Liu, and X. Du, "Deepqoe: Real-time measurement of video qoe from encrypted traffic with deep learning," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*. IEEE, 2020, pp. 1–10.

[13] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[14] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *PoPETs*, vol. 2019, no. 4, pp. 292–310, 2019.

[15] M. Korczynski and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 2014, pp. 781–789.

[16] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 2046–2059, 2021.

[17] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, K. Fu and J. Jung, Eds. USENIX Association, 2014, pp. 143–157.

[18] M. Shen, Y. Liu, L. Zhu, K. Xu, X. Du, and N. Guizani, "Optimizing feature selection for efficient encrypted traffic classification: A systematic approach," *IEEE Netw.*, vol. 34, no. 4, pp. 20–27, 2020.

[19] K. Abe and S. Goto, "Fingerprinting attack on tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[20] T. Shapira and Y. Shavitt, "Flowpic: Encrypted internet traffic classification is as easy as image recognition," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 680–687.

[21] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2004, June 10-14, 2004, New York, NY, USA*, E. G. C. Jr., Z. Liu, and A. Merchant, Eds. ACM, 2004, pp. 307–319.

[22] "The transport layer security (tls) protocol version 1.3," https://datatracker.ietf.org/doc/rfc8446/.

[23] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller, "Deep learning for time series classification: a review," *Data Min. Knowl. Discov.*, vol. 33, no. 4, pp. 917–963, 2019.

[24] "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[25] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[27] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*. IEEE, 2017, pp. 1578–1585.

[28] "Jd," https://www.jd.com.

[29] "Youtube," https://www.youtube.com/.