2ch-TCN: A Website Fingerprinting Attack over Tor Using 2-channel Temporal Convolutional Networks

Meiqi Wang*†‡, Yanzeng Li*†‡, Xuebin Wang*†‡, Tingwen Liu*†, Jinqiao Shi§* and Muqian Chen*†‡
* Institute of Information Engineering Chinese Academy of Sciences, Beijing, China
† National Engineering Laboratory for Information Security Technologies, Beijing, China
‡ School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
§ School of Cyberspace SecurityBeijing University of Posts and Telecommunications, Beijing, China

***Abstract*—In a website fingerprinting attack, an eavesdropper analyses the traffic between the Tor user and entry node of the Tor network to infer which websites the user has visited. Some recent work apply deep learning algorithms, however, most of them do not fully exploit the packet timing information. In this work, we propose a novel website fingerprinting attack based on a two-channel Temporal Convolutional Networks model that extracts features from both the packet sequences and packet timing information. Our attack is proved to perform better compared to the state-of-the-art attacks. Experiment results also show that the timing information is very useful for classification. Furthermore, we collect our own traffic traces between client and entry node, and transform them into three extraction layers: TCP, TLS and Tor cell layer, and meanwhile record Tor's cell log at the entry node. The experimental results show that the data of the cell layer is the most divisible among the three layers. Based on the experimental results, we conclude that the adversary at the entry node has an advantage over the one who just listens to traffic between client and entry node.**

***Index Terms*—website fingerprinting attack, deep learning, temporal convolutional networks**

I. Introduction

As people's growing awareness of privacy protection, in order to avoid censorship or eavesdropping, many network users tend to use privacy-aware communication tools that protect user anonymity, such as Tor [1]. Tor is one of the most popular anonymous communication tools and consists of volunteer nodes that relay encrypted traffic between the client and the web server, ensuring that no one node knows both the source and destination. Tor protects user privacy by separating the user's identity from his browsing behavior.

However, unfortunately, an attacker can still guess which specific websites a user has visited through traffic analysis from Tor's side channel [2]. A local, passive attacker can use the meta-information of the traffic collected between client and entry node of the Tor network, such as the size and direction of the packet in the traffic sequence, to infer the user's destination without decryption, which is known as Website Fingerprinting attack (WF attack). The attacker collects the client's observable traffic traces as sequences of packets. By classifying the client's packet sequences, the eavesdropper guesses which page the user is visiting.

Many studies have shown that website fingerprinting attacks are feasible and effective on Tor. Most prior attacks [2]–[6] use machine learning methods, that is, they extract manually selected features of traffic traces and then classify them with machine learning algorithms. However, manually character engineering is usually difficult and expensive, and the accuracy depends to a large extent on which features are extracted. Therefore, when facing defences that attempt to hide these particular features, these attacks may fail [5]. The work of Rimmer *et al.* [7] pioneered the use of Stacked Denoising Autoencoder (SDAE), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to automate feature selection, confirming the feasibility and superiority of applying deep learning to WF attacks. Since then, many work (such as [8]–[11]) have further studied how to improve the deep learning model to achieve higher accuracy.

Nevertheless, how to make better use of the various features hidden in the traffic is still an open question. Firstly, the importance of the time sequences has not been fully exploited for a long time. A number of prior studies such as [6], [7] didn't utilize the packet timing information. Hayes *et al.* [5] found that packet inter-arrival time statistics only slightly increased the attack accuracy. Secondly, how to process and parse the collected traffic traces is still an outstanding problem. Generally, researchers can extract the traffic data in three layers: TCP, TLS and Tor cell layer. Wang *et al.* [3] used the cell layer and removed SENDME cell to get the best classifier performance. According to Hayes *et al.* [5], their experiments showed no consistent improvements for classification from using one data layer over the other. Panchenko *et al.* [6] stated that the best classification accuracy is achieved by extracting data on the cell layer yet the differences are small and similar results can be obtained using even the most basic TCP layer.

Consequently, in this work, in response to the first question, we propose a novel and effective WF attack named 2ch-TCN, namely two-channel TCN. We utilize not only direction information but also time information of the cell sequences.

This work is supported by the Fundamental Research program of National Defence(JCKY2019211B001), Key Research and Development Program for Guangdong Province under grant(No.2019B010137003), the Fundamental Research Funds for the Central Universities (Grant no. 24820192019R-C56), National Defense Science and Technology Innovation Special Zone Project(18-H863-01-ZT-001-001-08)

Corresponding author: Xuebin Wang, Email:wangxuebin@iie.ac.cn

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

Experiment results show that our classifier can benefit greatly from time information. Our attack is proved to perform better compared to the state-of-the-art attacks and the time cost is acceptable. And in response to the second question, we collect our own traffic traces and transform them into the three extraction layers: TCP, TLS and Tor cell layer. Moreover, we collect direct cell logs by modifying Tor to record the basic information of each cell, and use the direct cell logs as ground truth to verify which layer provides the most information for website fingerprinting and whether having real cell information will give the attacker an advantage.

The key contributions of our work are as follows:

- We propose a new website fingerprinting attack using a two-channel TCN model that extracts features from both the packet direction and time information. We evaluate our attack on previous public datasets [4] [7], and compare to state-of-the-art approaches [6], [7], [9]. We achieve a better accuracy of 93.73% on the dataset of [4] and a better accuracy of 97.15% on the dataset of [7].
- We collect our own dataset and transform the traffic data in three extraction layers: TCP, TLS and Tor cell layer. We also record Tor's cell log at the entry node selected by the client during the visit. We make the generated dataset publicly available[1], allowing researchers to replicate our results and systematically evaluate new WF attacks.
- Based on the dataset we collected, we conduct WF attack experiments in three different data layers. The experimental results verify that the data at the cell layer can best benefit the classifier based on deep learning algorithms. We conclude that how to accurately extract cell sequence from traffic traces is still a problem that needs to be solved, and the adversary at the entry node with access to real cell information has an advantage over the one who just eavesdrop on traffic between the client and the entry node.

II. Background and Related Work

In this section we first present the threat model for website fingerprinting attack, and then we review the prior WF attacks on Tor, including machine-learning-based attacks and deep-learning-based attacks.

A. Threat Model

In this work, we adopt the same threat model as previous studies. The adversary is a local, passive eavesdropper who monitors the web browsing traffic between the victim and the Tor network, as shown in Fig. 1. A local attacker does not have to be physically close to the victim and he only has limited observation capacity. He can have access to the link between the victim and the entry node (like Adversary 1 in Fig. 1), or own the entry node the client chooses to connect to Tor (like Adversary 2 in Fig. 1). A passive adversary does not modify the traffic flow in any way and only records the packets

[1]The dataset can be found on the following URL: https://github.com/Meiqiw/2ch-TCN/.

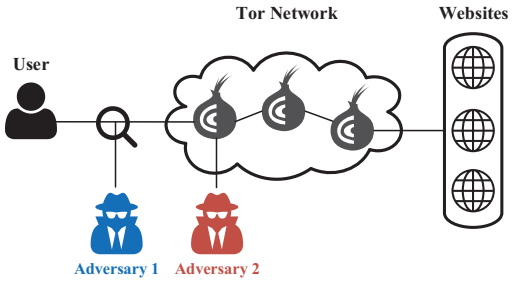

Fig. 1. The Threat Model of Website Fingerprinting Attack

transmitted during the web browsing process. Therefore, the adversary is hard to be detected. In realistic scenario, a website fingerprinting attacker can be a router, an Internet Service Provider (ISP), an Autonomous System (AS) operator, etc. To simplify the problem, we assume that the client loads a single website at a time without any other activity, such as file downloading.

In the WF attack, the adversary chooses some websites he interested in to monitor. He firstly visits each website several times and collects the traffic traces for each visit. Next, the attacker tries to build a website fingerprint for each website with the help of some classification algorithm. The collected traffic traces are labeled as their corresponding sites, and features of these traces are selected using manual or automated methods. The classifier is trained on these collected traces. Finally, the attacker applies the classifier to the new unlabeled traffic traces collected from the victim-initiated communications and tries to identify which website the client visits based on the output of the classifier.

B. Machine-learning-based WF Attacks

As Tor sends data in fixed-size (512-byte) Tor cells, attacks utilizing unique packet length (such as [12]) become fruitless on Tor. Thus, researchers turned to focus on features related to packet ordering ([2], [3], [13], [14]). For instance, Wang *et al.* [4] extracted a wide assortment of features including concentration of outgoing packets, bursts, etc. and utilized a k-Nearest Neighbor (k-NN) classifier, achieving 95% accuracy on 100 websites. Panchenko *et al.* [6] proposed CUMUL, using an SVM and mainly taking advantage of cumulative summations of packet sizes. This attack achieved similar accuracy to Wang *et al.*'s k-NN attack [4]. These attacks have become the state-of-the-art WF attacks and are used to benchmark other attacks.

These studies extract features of traffic traces manually mainly based on intuition, human experience and expert knowledge on how Tor and the HTTP protocol work. The manual feature extraction is expensive and difficult. Additionally, it makes the performance of the classifier largely dependent on the specific protocols or defenses. Therefore, when the protocol or defense policy alters to hide these extracted features, these attacks may fail.

It is worth noting that prior studies take more consideration on the packet sequence than timing information. Wang *et al.*'s k-NN attack [4] only added the mean and standard deviation

of the interpacket times into the 4,226-sized feature set. Hayes *et al.*'s *k*-FP attack [5], which used random forests as feature extractor and extracted 150 features from the traces, stated that packet inter-arrival time statistics only slightly increase the attack accuracy based on their assessment and ranking of the importance of features.

Besides, which layer provides the most divisible information for website fingerprinting is still an open question. Wang and Goldberg [3] used the cell layer and removed SENDME cells to get the best classifier performance. According to Hayes *et al.* [5], their experiments showed that using an extraction layer does not bring consistent improvement to the performance of the classifier over the other layers. Panchenko *et al.* [6] stated that the differences are small and using even the most basic TCP layer can obtain similar results to the cell layer, and the performance is not ideal when using data in the TLS layer.

C. Deep-learning-based WF Attacks

Recently, a few deep-learning-based WF attacks ([7], [8], [10], [11], [15]) have been proposed. Rimmer *et al.* [7] proposed three models: SDAE, CNN and LSTM and proved the feasibility of deep-learning-based website fingerprinting. Sirinam *et al.* [8] presented Deep Fingerprinting (DF), using a CNN model with a sophisticated architecture design and achieving promising results on their own datasets, and it became current state-of-the-art WF attack. All of the above work utilized the packet direction information of the cell layer, while not fully exploited the time information of the packet. Bhat *et al.* [9] designed Var-CNN, a complicated model using both the directional and timing information of the traffic traces. Corresponding to the above two sequences, they trained two optimized CNNs, named the direction CNN and the time CNN, and then ensembled them to achieve higher accuracy. They claimed that their attack worked well with smaller training sets. While they didn't get rid of the reliance on manual extraction features. That is, they used 7 artificially extracted features (such as the total number of packets, the total transmission time, and so on) which they called metadata so as to improve the performance of their model.

III. Our 2ch-TCN Model

In this section we first introduce a short background on temporal convolutional networks, and then present details of our two-channel TCN model.

A. Temporal Convolutional Networks

CNN is a specific architecture of neural networks, widely used for natural language processing tasks, text classification tasks, and sequence classification tasks. Its convolution operations extract high-level statistical characteristics from fragments of the input sequence. Some variants of CNN could present different feature and be suitable for specific tasks. For example, causal convolutions could capture the ordering feature of sequence context without recurrent connection like RNN. Another classic variant is dilated convolutions [16]. It

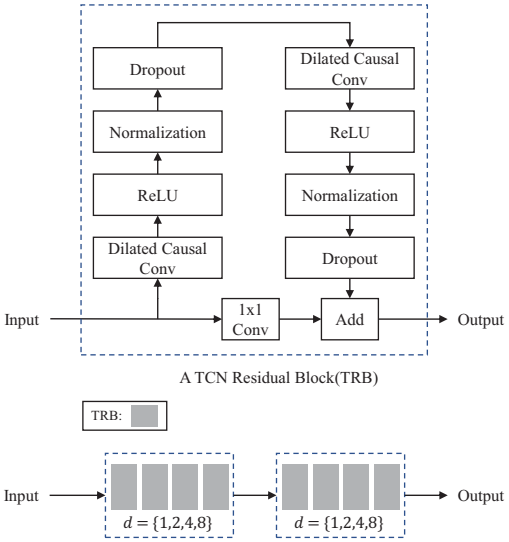

Fig. 2. Temporal Convolutional Networks

expands the alignment of the kernel weights by dilation factor, increasing kernel size and gaining a large receptive field [17].

Based on CNN and inspired by WaveNet [18], Bai *et al.* [19] proposed TCN for sequence modeling, combining up dilated convolutions [16] and causal convolutions, using 1D Fully-Convolutional Network (FCN) [20] and residual block [21] architecture. To compose a TCN model, Bai *et al.* stack dilated causal convolution layer, weight normalization layer [22], spatial dropout [23] and Rectified Linear Unit (ReLU) [24] into a residual block, then stack multiple residual blocks vertically. Fig. 2 shows a fundamental structure of TCN. Combining up advantages of its components, TCN shows high performance of parallelism, flexible receptive field size, stable gradients and low memory requirement for training. Moreover, it outperforms prior to state-of-the-art methods in many sequence modeling tasks over open datasets. Because it can better extract the sequence information of the data compared with ordinary CNN, and can better learn the characteristics of long sequences compared with Recurrent Neural Network (RNN).

B. Details of 2ch-TCN

We model WF attack task as a special sequence classification task. With taking two kinds of sequence (direction or size of packet/cell sequence and timing sequence) as input, the model needs fitting and learning the high-level features of sequence, then predict the catalogue of an unknown sequence pair (which means a successful attack). For WF attack task, we introduce a TCN-based model architecture, with two inputting channels. As an excellent sequence model, TCN could extract sequence ordering feature and high-level characteristics both from direction sequence and timing sequence, and build representation vector for specified task. We use fully-connected layer to collect and combine the output of direction and timing sequence representation model, then use a softmax layer to map traffic sequence representation to the corresponding class labels. To avoiding overfitting, we also use dropout mechanism

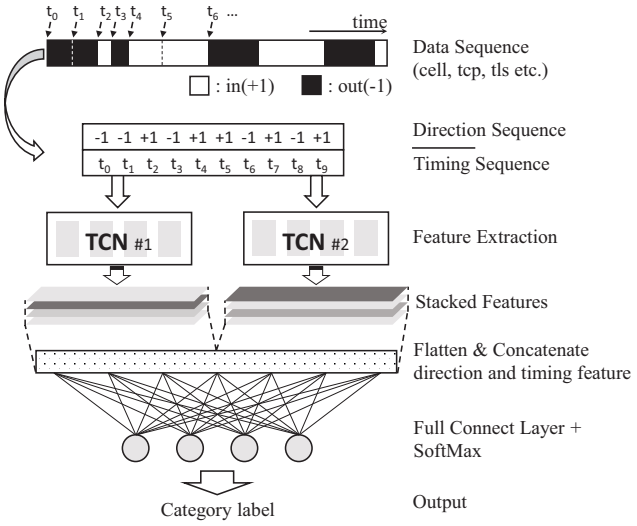

Fig. 3. Architecture of Our Model

[23] for the last dense layer. According to Hayes *et al.* [5] and Bhat *et al.* [9], timing sequence should have been insignificant. Benefit from TCN's advantages, especially from its stable gradients, timing sequence representation could be trained efficiently with direction sequence at the same time, which promotes each other's convergence of gradient descent. And due to the sequence feature extraction capability of TCN, our model could gain outstanding results without any artificial feature or metadata. The schematic architecture of the proposed model is shown in Fig. 3.

As shown in Fig. 3, our model architecture is mainly composed of TCN module. The direction model and timing model share the same configurations. In order to fit large-scale dataset as much as possible, we use 16 layers of stacks, one of which is shown in Fig. 2. Then we set 16 filters with 8 size of kernel for each convolution layer in our model. For dilated causal convolution layers, we provide a dilation factor list: {1,2,4,8}, thus those layers would follow the list to set dilation factor. Benefit from TCN's superiority in capturing ordering sequence features and preventing memory leaking, we pad and truncate sequence into 5000 length. For training, we use categorical cross-entropy as loss function and Adam [25] as optimizer, with 0.001 learning rate.

IV. Experimental Setup

In this section, we introduced datasets we used and the steps for data collection and data processing.

A. Data Collection

Most of publicly available datasets provide packet sequence information in the cell layer. In order to obtain the timing information of the traffic traces and the complete traffic in TCP layer and TLS layer, we collect data to form our own dataset for experiments. In order to collect data more efficiently, we use a distributed setup, utilizing 20 Virtual Machines (VMs) on cloud environment provided by Vultr[2]. These virtual machines are located in different countries including Japan, Singapore, Australia, Germany, France, the Netherlands and the United States, so as to ensure the diversity of traces. Each VM is configured with 2 CPUs and 4GB of RAM. On each VM, 10 dockers are deployed, and each docker has a separate Tor process (version 0.3.3.7). To access the Tor network, we use `Selenium`[3] (version 3.12.0) to control headless browser `Firefox` (version 60.0.2), utilizing a SOCKS5 proxy listened by Tor. We recorded the traces of web pages leveraging `tcpdump`[4]. Web pages are given 60 seconds to load, and upon loading the page, it was left open for an additional 3 seconds, after which the browser is closed and the Tor process is killed. Next, `tcpdump` and Tor process are restarted. After waiting for another 3 seconds, the browser is restarted and ready to visit the next website. With this setup, a new circuit is established each time the client visits a website. We also perform page loading with no caches and time gaps between multiple loads of the same web page as Wang and Goldberg [3] recommended.

While recording the traffic trace, we also record cell logs on the entry node chosen by the client by modifying Tor to record the basic information of each cell, including the direction and cell command, and use the direct cell logs as ground truth to verify whether the cell layer can provide more information for WF attack and whether having access to real cell information will give the attacker an advantage.

It is worth mentioning that we increase the randomness of the data by maximizing the differences in the web browsing environment. Our data collection method ensures that a new circuit is established for each visit, and each traffic trace contains the process of circuit construction. The circuit diversity of our dataset is more abundant than traces in the realistic scenario where the adversary implements a targeted attack on a single victim. Using our data for training makes the model more universal, making the attack applicable to any Tor user. Undoubtedly, this will increase the difficulty of WF attacks compared to using the previous idealized data collection method. We also hope to observe the performance of our attacks and the state-of-the-art attacks under such unfavorable conditions for attackers.

B. Dataset

In total, we evaluate our attack in comparison with state-of-the-art methods on three different datasets: Wang *et al.*'s *k*-NN dataset [4] (referred to as *Wang14*), Rimmer *et al.*'s dataset [7] (referred to as *Rimmer18*) and our own dataset (referred to as *MultiLayer20*). In this subsection we describe the details of these datasets and explain why we choose them.

1) Wang14: The *Wang14* dataset contains 100 websites with 90 instances for a closed-world evaluation. These 100 monitored pages was compiled from a list of blocked websites in China, the United Kingdom, and Saudi Arabia. This dataset has been well studied by many prior researches [4]–[6], [15].

[2]http://vultr.com

[3]http://www.seleniumhq.org/

[4]http://www.tcpdump.org/

Traces in *Wang14* are cell sequences, including information about the direction of each cell and the timestamp when each cell is captured. They collected data in batches, tried to use each circuit as long as possible, and maintained the same circuit in one batch. In addition, they used one client located at a fixed IP to collect all the traces.

2) Rimmer18: The *Rimmer18* dataset consists of 900 websites, with 2,500 valid network traces each. Their monitored websites are sourced from the homepage of 1,200 of the most popular websites according to Alexa [26]. *Rimmer18* is the largest WF dataset ever gathered to date, and it was collected with a distributed setup by 240 worker threads of 15 VMs. The collection was divided into four iterations, with each iteration split up into 30 batches. The traces in *Rimmer18* are also from cell layer, but only contains the direction of each cell without any timing information. In order to facilitate comparison with *Wang14*, we use a subset of *Rimmer18* containing 100 websites with 2,500 instances each.

3) MultiLayer20: Following our data collection method, we collected up to 3,000 network traces of 2,000 websites (from Alexa Top websites and Tor hidden services[5], each with 1000 websites). Similar to the previous work, after data collection, we filtered out invalid traces and outliers, which caused by timeout or crash of the browser or Selenium driver. Then we removed websites with a high amount of invalid traces from our dataset. Finally, we balanced our dataset to ensure every site has the same number of traces. After completing the above operation, we eventually obtained *MultiLayer20*, consisting of two subsets of monitored web pages: (i) 2,300 instances from each of the 500 top Alexa monitored web pages and (ii) 1,400 instances from each of 850 popular Tor hidden services. Our dataset contains both direction and time information of each packet. Besides, we also transform the traffic data in three extraction layers: TCP, TLS and Tor cell layer, which we will explain in detail in the next subsection.

C. Data Extraction and Processing

As shown in Fig. 4, at the application layer, Tor embeds the encrypted data into a fixed-size (512-byte) packet, which is called a cell. And the cell is further embedded into the TLS record. Multiple cells can be grouped into a single TLS record. Finally, in the transport layer, TLS records are typically fragmented into multiple TCP packets, the size of which is limited by the Maximum Transmission Unit (MTU). Note that several TLS records can be within a single TCP packet.

In order to observe the performance of various methods at different data layers, we parse the original captured pcap files into TCP packet sequences and TLS record sequences, with the help of `Tshark`[6] (version 1.12.1). Then we extracted cell sequences from the TLS record sequences following the method proposed by [3]. Serving as ground truth, the cell logs were converted into the same form as the cell sequences. In Section V-B we provide an evaluation of the different layers of extraction and discuss their impact on classification.

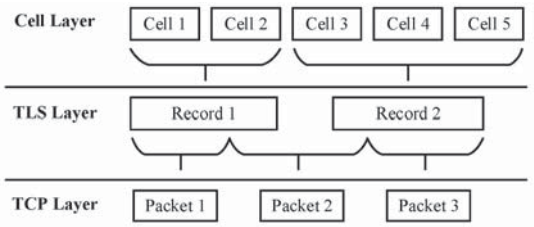

Fig. 4. Layers of Data Transport in Tor

V. Evaluation and Discussion

In this section we evaluate our WF attack 2ch-TCN in Tor. In Section V-A, we compare our approach to state-of-the-art attacks on previous public datasets and show that our method is superior in classification accuracy. By comparing the accuracy, loss and evaluation time of several attacks with a growing number of training epochs, we show that the packet time information not only helps the classifier achieve higher accuracy, but also improves the convergence speed of the model. In Section V-B, we reevaluate prior work and our attack on our dataset. By comparing the performance of attacks in the three data extraction layers, we conclude that the cell layer does benefit the classifier and performs better than the other two layers.

A. Comparison with State-of-the-art Attacks

In this subsection, we compare the performance of our novel attack to four state-of-the-art approaches: CUMUL proposed by Panchenko *et al.* [6], SDAE and CNN proposed by Rimmer *et al.* [7] (In order to avoid confusion with common neural network models, we will refer to them as Rimmer-SDAE and Rimmer-CNN respectively), and Var-CNN proposed by Bhat *et al.* [9]. We choose these attacks for comparison because the CUMUL classifier has been proven by [7] to be superior to previous machine-learning-based methods, and the other three methods are most representative deep-learning-based methods at present with promising performance.

We implement our model with Keras [27], which using Tensorflow[7] as backend. We practise deep learning experiments on a platform that has two Nvidia Tesla P100 GPUs with 24 GB GPU memory, which can make high parallelism performance of CNNs due to cuDNN primitives [28]. We use the default parameter settings and perform each experiment three times and present the results as an average.

Among the above four state-of-the-art methods, only Var-CNN utilizes time information. We adopt a single-channel TCN model (referred to as TCN-dir) that only makes use of the direction information and correspondingly, we choose the direction CNN of Var-CNN (referred to as Var-CNN-dir) for comparison. We first compare the performance using the *Wang14* dataset and the *Rimmer18* dataset. For *Wang14*, we use 60 instances of each website for training and 30 instances

[5]A hidden service is a website users visit uses Tor technology to stay secure and anonymous, which is accessed through its onion address. As the prior work, we chose hidden services based on the list provided by the .onion search engine http://www.ahmia.fi/.

[6]https://www.wireshark.org/docs/man-pages/tshark.html

[7]https://www.tensorflow.org/

TABLE I
ACCURACY(%) OF THE FIVE ATTACKS ON *Wang14* AND *Rimmer18*

	Wang14	***Rimmer18***
CUMUL	91.38[a]	95.43[b]
Rimmer-CNN	71.43	91.23
Rimmer-SDAE	87.78	95.50
Var-CNN-dir	93.20	97.01
Var-CNN	93.33	-
TCN-dir	92.60	**97.15**
2ch-TCN	**93.73**	-

[a]Based on Experiment Results of [6].
[b]Based on Experiment Results of [7].

for testing. And for *Rimmer18*, we use 2,200 traces of each website for training and 300 traces for testing. In this scenario, the accuracy is a suitable metric to measure the performance of classifiers, which is defined as the proportion of correct classifications (true positive and true negative) among the total number of traces identified. So we use the classification accuracy to define the effectiveness of the attacks.

The results for both datasets are shown in TABLE I. As we can see, with the number of traces per site increasing, all classifiers' performance is improved. This shows that these attacks are practically feasible for larger scale of traffic data. Moreover, collecting more traffic traces for each site is beneficial to the classification accuracy of these classifiers, as more adequate data can help the models understand the features of packet sequences more thoroughly. When the number of traces of each website is small, Rimmer-CNN and Rimmer-SDAE do not perform well. Consistent with the experimental results in [7], we observed significant improvements in these two attacks when amount of traces grows. We conclude that their models have a strong dependence on the amount of data. In contrast, the machine-learning-based CUMUL, the deep-learning-based Var-CNN and our method can still perform well even when the number of instances on each website is limited. Compared to Var-CNN, our model performance is more affected by the number of instances of each website. Our model outperforms than state-of-the-art approaches when there are enough traffic traces on each website.

Since *Rimmer18* does not contain time information, we only use the dataset of *Wang14* to compare the result of Var-CNN with that of our model. It can be seen that after using the time information, the accuracy of our model increases far more greatly than Var-CNN, from 92.60% to 93.73%. However, the Var-CNN model does not benefit much from the time sequences, showing the accuracy of 93.30% with an increase of only 0.1%. This shows that our model can better learn the features of time sequences, so that the gain brought by utilizing time information is greater. We conclude that when taking the packet time information into consideration, our model stands out and shows the highest accuracy compared to state-of-the-art attacks including machine-learning-based methods and deep-learning-based methods.

Next, we compare the accuracy and loss of our model and the most promising state-of-the-art approach, Var-CNN, with a growing number of training epochs. The result is shown in Fig. 5. Obviously, our model can rapidly improve accuracy and reduce loss in the early stage of training. We conclude that our model converges much faster than Var-CNN, which improves the efficiency of training.

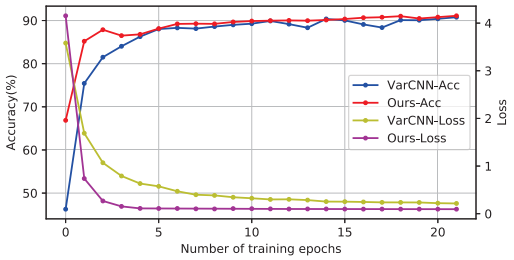

Fig. 5. Convergence Speed of Our Model and Var-CNN on *Wang14*

It is worth noting that Var-CNN uses two CNN models and ensembles them, in which they need to train one CNN after another. This undoubtedly increases the time cost of training, with a tiny increase in accuracy. By comparison, our model only needs to be trained once. In addition, Var-CNN still relies on manual extraction of metadata, while our attack relies entirely on neural networks to automatically extract features. Therefore, we can draw a conclusion that our model can better learn the characteristics of the traffic traces with the help of both the packet direction information and timing information.

B. Experiment Results Using Our Dataset

In this subsection, we evaluate three most promising deep-learning-based attacks: Rimmer-SDAE, Var-CNN-dir and TCN-dir on our dataset *MultiLayer20*. For comparison purposes, we use models that do not use time information. For *MultiLayer20*, we use 2,000 instances for training and 300 instances for testing. Our dataset consists of traffic traces in three extraction layers: TCP layer, TLS layer and the cell layer, in addition to the cell log recorded at the same time, so we can find out which layer is most useful for WF attack through experiments, and compare the difference between cell sequences extracted from TLS records by Wang's method [3] (referred to as Wang's cell) and real cell sequences in experimental results.

The results are shown in TABLE II. As we can see, the classification of all attacks has dropped. The reason is as mentioned above: in the process of collecting data, we ensure that a new circuit is established for each visit, which increases the randomness of the data. Our classifier still performs well compared to the other two classifiers, achieving the highest accuracy of 86.31% in the cell layer, and in the other two layers, our model performance is close to Var-CNN, although no time information is used. This implies that our model is very stable and can handle the situation with a wide variety of circuits in traffic traces.

By comparing the results of these three approaches using data in different extraction layers, we observed that the data from the cell layer does yield the best results. Theoretically, the Tor cell is a more consistent and basic unit than TCP packet and TLS record. Thus, the cell layer can provide

TABLE II
ACCURACY(%) OF ATTACKS ON *MultiLayer20* IN THREE LAYERS

	TCP	TLS	Wang's Cell	Real Cell
Rimmer-SDAE	44.87	41.97	70.66	79.28
Var-CNN-dir	76.35	67.61	77.59	85.72
TCN-dir	73.87	67.41	79.78	**86.31**

more divisible data for the classifiers. The worst classification accuracy is achieved by extracting data on the TLS layer, which is consistent with the experimental results of [6].

Another important finding of the experiment is that when using the real cell layer, the classification accuracy of all approaches is significantly improved by around 8%. This indicates that cell sequences converted by means of [3] may have lost some characteristic information compared to real cell sequences. As a result, how to accurately extract cell sequence from traffic traces is still a problem that needs to be solved. Meanwhile, we draw the conclusion that the adversary at the entry node with access to real cell information has an advantage over the one who just eavesdrop on traffic between the client and the entry node.

VI. CONCLUSION AND FUTURE WORK

In this work, we propose a novel website fingerprint attack, 2ch-TCN, that extracts features automatically from both the packet direction and time information. Experiment results show that our classifier can benefit greatly from time information. Our attack is proved to perform better compared to the state-of-the-art attacks with an acceptable time cost. Besides, we collect our own traffic traces and transform them into the three extraction layers. We conclude that the data of the cell layer is the most divisible among the three layers. Another interesting finding is that the attacker at the entry node with access to Tor's real cell information can have a great advantage over the one who just eavesdrops on the client and the entry node.

We see a bright future for applying deep learning to WF attacks. We will further improve our model to increase the classification accuracy and apply it to more complex scenarios, such as where users accessing multiple websites at the same time. At the same time, how to defend against deep-learning-based attacks is also a question worth pondering. Besides, how to make the data extracted from the traffic traces closer to the real cell is also an urgent problem to be solved.

REFERENCES

[1] P. Syverson, R. Dingledine, and N. Mathewson, "Tor: The secondgeneration onion router," in *Usenix Security*, 2004.

[2] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[3] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013, pp. 201–212.

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

[5] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique." in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[6] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.

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

[8] P. Sirinam, M. Imani, M. Juarez, 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*, 2018, pp. 1928–1943.

[9] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, pp. 292–310, 10 2019.

[10] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1131–1148.

[11] T. Pulls and R. Dahlberg, "Website fingerprinting with website oracles," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 1, pp. 235–255, 2020.

[12] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 31–42.

[13] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 332–346.

[14] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[15] 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.

[16] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[17] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka, "Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1442–1450.

[18] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio." in *SSW*, 2016, p. 125.

[19] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] "Alexa the web information company," http://alexa.com, accessed June 2018.

[27] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[28] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *Computer Science*, 2014.