Proceedings of the APAN – Research Workshop 2016
ISBN 978-4-9905448-6-7

Fingerprinting Attack on Tor Anonymity using Deep Learning

Kota Abe and Shigeki Goto

***Abstract*— Tor is free software that enables anonymous communication. It defends users against traffic analysis and network surveillance. It is also useful for confidential business activities and state security. At the same time, anonymized protocols have been used to access criminal websites such as those dealing with illegal drugs. This paper proposes a new method for launching a fingerprinting attack to analyze Tor traffic in order to detect users who access illegal websites. Our new method is based on Stacked Denoising Autoencoder, a deep-learning technology. Our evaluation results show 0.88 accuracy in a closed-world test. In an open-world test, the true positive rate is 0.86 and the false positive rate is 0.02.**

***Index Terms*— Network Security, Tor, Fingerprinting Attack, Deep Learning, Autoencoder**

I. INTRODUCTION

The Onion Router (Tor) is free software that enables anonymous communication. [1, 2]. It defends users against traffic analysis and network surveillance. It is also useful for confidential business activities and state security. At the same time, anonymized protocols have been used to access criminal websites such as those dealing with illegal drugs. There is a need to develop a method that can identify websites when anonymized protocols are used.

This paper proposes a new method for launching a fingerprinting attack to analyze Tor traffic in order to detect users who access illegal websites. Using a fingerprinting attack, we can identify a website that a user accesses on the basis of traffic features such as packet length, number of packets, and time. We can analyze this information from captured packets regardless of encryption. Our new method for fingerprinting attacks is based on Stacked Denoising Autoencoder (SDAE), a deep-learning technology. Our evaluation results show 0.88 accuracy is in a closed-world test. In an open world test, the true positive rate (TPR) and false positive rate (FPR) are 0.86 and 0.02, respectively.

The remainder of this paper is organized as follows. Section II explains the technical background. Section III describes related work. Our new method is proposed in Section IV. Section V shows the evaluation results. Section VI concludes the paper.

Kota Abe and Shigeki Goto are with the Department of Computer Science and Engineering, Waseda University, Shinjuku, Tokyo 169-8555 Japan
e-mail: (see http://www.goto.info.waseda.ac.jp).

II. TECHNICAL BACKGROUND

A. *Tor Anonymity*

Tor [1, 2] is a popular anonymized protocol. Figure 1 shows an example of a Tor configuration. At the initial setting, there are three nodes between a user and a web server, as shown in Figure 1. Tor traffic data is encrypted using Transport Layer Security (TLS) between a user and each Tor node. Thus, Tor nodes do not know the original plain data, with one exception. The closest node to the web server can read the original data without encryption. In a Tor configuration, each node knows only the Internet Protocol (IP) addresses of adjacent nodes that are directly connected to the node.

In the Tor protocol, content data is encapsulated into a series of *cells*, each with a fixed length of 512 bytes. It is difficult to estimate the original content only from the packet length.

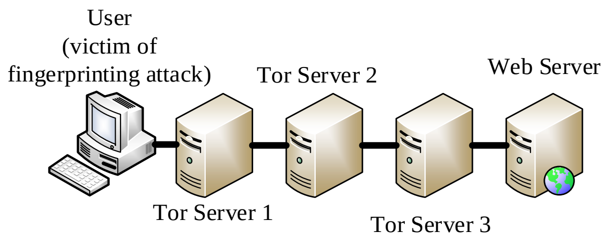

Fig. 1. Configuration of Tor

B. *Fingerprinting Attacks on a Website*

1) *Fingerprinting*

A website fingerprinting attack aims to detect a website even if the traffic is encrypted using Tor or a virtual private network (VPN). We cannot specify the website by inspecting the encrypted payload. However, we can utilize the packet information, such as packet length, number of packets, and time. In a fingerprinting attack, we can specify a website by providing the packet information.

There are two methods for capturing traffic data in Tor. In the first method, an attacker (analyzer) prepares an entry node of Tor and captures the traffic through this node. However, the Tor protocol selects nodes at random. It is unlikely that a specific victim connects to the attacker's node. In the second method, an attacker (analyzer) is a network operator, such as an Internet service provider (ISP). He or she can capture traffic packets between a victim and the entry node of Tor. This is a realistic scenario. This paper proposes a new approach using the second method.

2) Closed- and Open-World Tests

There are two evaluation schemes for fingerprinting attacks. The first scheme is a *closed-world* test. It conducts a test in which a victim can access only a limited number of websites, which the attacker attempts to detect. For example, an attacker might prepare 100 monitored sites and investigate the features of these 100 websites. The victim can access only these 100 websites.

The second scheme is an *open-world* test. In such a test, a victim can freely access any websites on the Internet. The attacker must be able to determine whether a website is monitored or non-monitored. If it is a monitored website, the attacker must be able to determine which website among the 100 monitored sites it is. This paper uses two evaluation schemes, closed and open.

C. SDAE

Deep learning is an attractive method in machine learning. It is called *deep* because it utilizes a multiple-layered neural network. An autoencoder is a deep-learning technique. This paper uses SDAE.

An autoencoder is a neural network that consists of input, hidden, and output layers. Figure 2 shows an example of an autoencoder. It calculates weights on directed edges in Figure 2 by learning from input data. One specific autoencoder feature is that the input data (vector) and the output data (vector) must be equal.

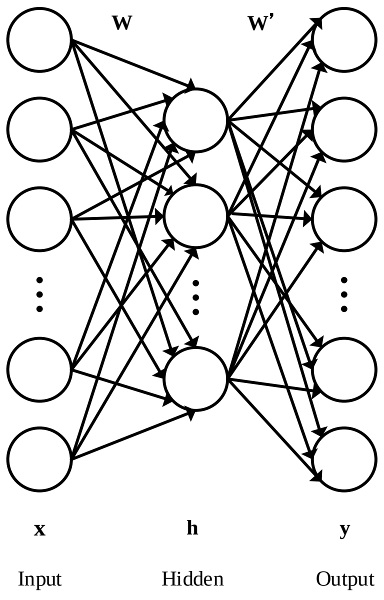

Fig. 2. Structure of an Autoencoder

An autoencoder is represented by a mathematical formula. In formula (1), the input layer is represented as a vector $\boldsymbol{x}$, the output of the hidden layer as a vector $\boldsymbol{h}$, and weights from the input layer to the hidden layer as a matrix $\boldsymbol{W}$ and vector $\boldsymbol{b}$. The vector $\boldsymbol{b}$ represents bias terms. We also define an activation function f. Data propagation from the input layer to the hidden layer is calculated using formula (1).

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{1}$$

Similarly, we define the output from the output layer as a vector $\boldsymbol{y}$, and the weights from the hidden layer to the output layer are represented as a matrix $\boldsymbol{W'}$ and vector $\boldsymbol{b'}$. The vector $\boldsymbol{b'}$ consists of bias terms. We also define an activation function f'. Data propagation from the hidden layer to the output layer is calculated using formula (2).

$$\boldsymbol{y} = f'(\boldsymbol{W'}\boldsymbol{h} + \boldsymbol{b'}) \tag{2}$$

The autoencoder determines the weights $\boldsymbol{W}$ and $\boldsymbol{W'}$ that equalize the input $\boldsymbol{x}$ and output $\boldsymbol{y}$. The weights are calculated using formula (3), which minimizes the difference between the input data $\{\boldsymbol{x_i}, \ldots\}$ and output $\boldsymbol{y}$.

$$\min_{\boldsymbol{W},\boldsymbol{b},\boldsymbol{W'},\boldsymbol{b'}} \sum_i \|\boldsymbol{x_i} - f'(\boldsymbol{W'}f(\boldsymbol{W}\boldsymbol{x_i} + \boldsymbol{b}) + \boldsymbol{b'})\|_2^2 \tag{3}$$

Using an autoencoder, we can decrease the dimensions of data vectors. The dimension of $\boldsymbol{h}$ is less than that of $\boldsymbol{x}$ or $\boldsymbol{y}$. The output vector $\boldsymbol{h}$ of the hidden layer is used as a feature vector in machine learning.

We can combine multiple autoencoders by overlapping a hidden layer as an input of the second autoencoder. This type of autoencoder is called Stacked Autoencoder (SAE). Figure 3 shows an example.

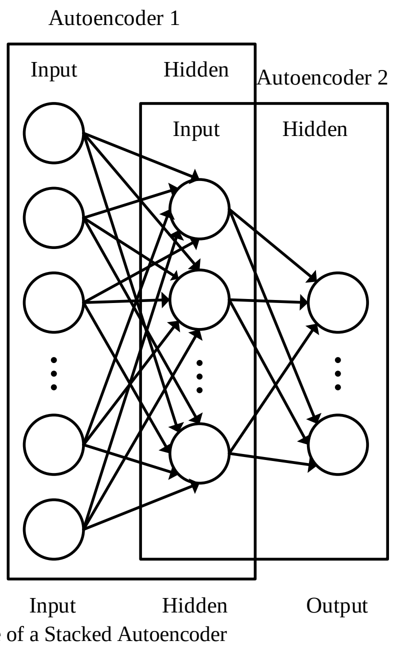

Fig. 3. Structure of a Stacked Autoencoder

It can be meaningful to add *noise* to an input vector. This type of autoencoder is called a denoising autoencoder (DAE). By adding noise data, an autoencoder can avoid overlearning or overfitting, with the result that formula (3) is satisfied only for the training data. Noise is sometimes useful to generalize the training data. A DAE can attain higher accuracy.

We can further combine multiple DAEs similarly to SAEs, This type of autoencoder is called SDAE. This paper uses SDAEs. We use Pylearn2 software [3] as a deep-learning tool.

III. RELATED WORK

A. Optimal String Alignment Distance (OSAD)

In 2013, Wang and Goldberg [4] conducted a fingerprinting attack using OSAD. In their method, a sequence of Tor cells is treated as a string. If two instances of a cell string are captured for the same site, the distance between the two instances is small. If they are captured for two different sites, the distance of the two instances is large. Wang and Goldberg used OSAD in an algorithm to calculate the distance.

Wang and Goldberg used this distance as the *kernel matrix* in a support vector machine (SVM). They defined the distance and the kernel by formulas (4) and (5), respectively. s_1 and s_2 are two strings, and the distance between s_1 and s_2 is $\mathrm{D}(s_1, s_2)$.

$$\mathrm{D}'(s_1, s_2) = \frac{\mathrm{D}(s_1, s_2)}{min(|s_1|, |s_2|)} \tag{4}$$

$$\mathrm{K}(s_1, s_2) = e^{-\mathrm{D}'(s_1, s_2)^2} \tag{5}$$

When $\mathrm{D}' = 0$, two strings are equal, and K becomes one. When the distance between two strings is large, K becomes small. When $\mathrm{D} \rightarrow \infty$, the limit of K becomes zero. Therefore, we can use K as the kernel matrix of an SVM. Wang and Goldberg used the one-against-one method in their SVM. This method is used for multi-class classification by repeating two-class classifications and by performing majority voting.

B. k-Nearest Neighbor Algorithm (k-NN)

In 2014, Wang et al. [5] proposed another fingerprinting attack using the k-nearest neighbor (k-NN) algorithm. In their new method, they extract features from captured packets.

- General features (total transmission size, total transmission time, and numbers of incoming and outgoing packets)
- Packet ordering
- Concentration of outgoing packets
- Bursts

Some features are more meaningful than others. Then, they determine the weights of features. Finally, they classify test data using the k-NN method with features and weights.

IV. NEW METHOD

A. Dataset for Learning and Evaluation

This paper uses the same dataset as that of Wang [6] in our evaluation experiment. This dataset contains 100 sites as *monitored* web sites and 9,000 sites as *non-monitored* sites. Monitored sites are used in the closed-world test. Non-monitored sites are used in the open-world test. Each monitored site has 90 instances (cells), and each non-monitored site has one instance. Monitored sites consist of porn sites, Bit Torrent trackers' sites, and sites that have religious or political contents. Access to these sites is blocked in China, United Kingdom, and Saudi Arabia. Non-monitored sites consist of Alexa's list [7], which covers ordinary popular web pages. In Figure 4, the first column records when a cell is captured. The timestamp unit is seconds. The time at which the first cell is sent is 0.0. The second column indicates the direction of a cell. When a cell is sent from a victim (target) to a Tor node, it is represented as 1. When a cell is sent from a Tor node to a victim, it is represented as -1. This time sequence starts when the web page begins loading and ends when the last cell is sent.

0.0	1
0.0	1
0.116133928299	1
0.499715805054	-1
0.499715805054	-1
0.782404899597	-1
0.969846963882	-1
0.969846963882	-1
0.969846963882	-1
0.969846963882	-1

Fig. 4. Example of dataset.

We count the number of cells in a packet. Since the size of a cell is fixed at 512 bytes, the number of cells is counted by dividing the packet length by 600. We use not 512 but 600 because we consider inter-cell headers and the overhead [10] .Tor sends cells for flow control at regular intervals. Such a control cell is called a SENDME cell. SENDME cells are not useful in fingerprinting attacks. We exclude SENDME cells from the dataset.

B. Proposed Method

First, an attacker (analyzer) collects training data for machine learning. The attacker accesses websites he or she wants to monitor through Tor fingerprinting and then captures the traffic data repeatedly, e.g.,., 100 times. The attacker also collects traffic data from a large number of other websites. The data is used for the open-world test. Since this paper uses Wang's dataset, we can omit the data collection phase.

Next, the attacker extracts Tor cells from the captured data. These are used as input to the autoencoder. Again, we can omit this phase, because we use the same dataset as that in Wang's method. Tor cells are already extracted. Then, we sort out data to create an input vector to the autoencoder. This paper uses the direction of a cell as an element of an input vector. It is a simple method. We do not use other features. It should be noted here that input vectors have a fixed length (dimension). The original traffic data have a variety of lengths (dimensions) according to the traffic pattern. We truncate a sequence of cells if the length is greater than 5,000. If the number of cells is less than 5,000, we put 0 as dummy padding to create a vector of size 5,000. Figure 5 shows an example of input data corresponding to Figure 4.

1, 1, 1, −1, −1, −1, −1, −1, −1, −1, −1,
−1, −1, 1, −1, 1, 1, ... , 0, 0, 0, 0, 0

Fig. 5. Example of input data

After preparing the data, the attacker conducts training using the SDAE. In our experiment, we specifically use a multilayer perceptron (MLP) that has two layers of SDAEs and an output layer realized by a *softmax* function. The parameters of the SDAE and MLP will be shown in the next section (V). Before inputting the training data, we randomize the order of the training vectors in the data set. If a *batch* has many similar vectors, the efficiency of learning might be decreased.

The *test* data for evaluation is prepared similarly to the *training* data.

V. Evaluation

A. Environment

Table 1 shows the experimental environment. We use Compute Unified Device Architecture (CUDA) [8] to accelerate the training using a graphical processing unit (GPU). Table 1 shows the machine specification, which includes a GeForce GTX 750 Ti graphics card by NVIDIA.

TABLE 1
ENVIRONMENT OF EXPERIMENT

OS	Ubuntu 14.04.03 LTS
CPU	Intel Core i7-4790
RAM	32 GB
GPU	NVIDIA GeForce GTX 750 Ti

B. Closed-World Test

1) Overview

In the closed-world test, the dataset contains 100 monitored sites, with each site containing 90 cell instances. Seventy-two instances are used for training data, and 18 instances for test data. This closed-world test is a multi-class classification. We labeled monitored websites as class 0 to class 99.

2) Layer

We used an MLP with two layers of SDAEs and with the output layer realized by a softmax function. Parameters of Pylearn2 are shows in Tables 2 and 3.

Nvis and *nhid* are the dimensions of the input and hidden layers of the Autoencoder, respectively. Learning rate is a coefficient during the weight-training phase.

TABLE 2
PARAMETERS OF SDAE (CLOSED-WORLD TEST)

Parameter	First Layer	Second Layer
nvis	5000	500
nhid	500	125
learning_rate	0.001	0.001
batch_size	50	50

TABLE 3
PARAMETERS OF MLP (CLOSED-WORLD TEST)

Parameter	MLP
Nvis	5000
n_classes	100
learning_rate	0.005
batch_size	200

3) Results

We conducted a series of closed-world tests while changing the number of learning sessions (*max_epoch*) every five times. Values of *max_epoch* in each DAE and the output layer are the same. Figure 6 shows the accuracy of the closed-world tests. The highest accuracy of 0.88 is attained when the number of learning sessions is 50.

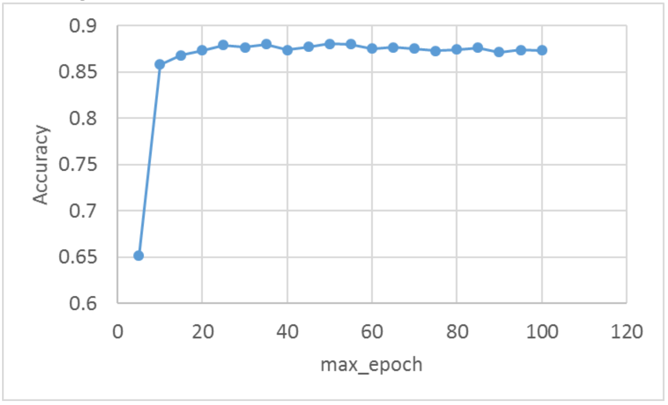

Fig. 6. Relation between max_epoch and accuracy in closed world test.

We also conduct a series of closed-world tests by changing the dimensions, *nvis* and *nhid*. We fix the *max_epoch* value as 50. The results are shown in Table 4. There is no major change in the accuracy by the dimension parameters of the hidden layer of the SDAE. The maximum accuracy is attained when the *nhid* values of the first and second layers are 1,000 and 500, respectively. When the *nhid* values are 500 for the first layer and 125 for the second, the results are similar.

TABLE 4
RESULTS WHEN CHANGING NVIS AND NHID
(CLOSED-WORLD TEST)

Accuracy		1st layer			
		250	500	750	1000
2nd layer	125	86.4	88.1	86.9	86.9
	250	87.1	87.2	87.6	87.6
	500	-	87.3	87.2	88.2
	750	-	-	87.9	87.3
	1000	-	-	-	87.6

4) Execution time

Table 5 shows the execution time when *max_epoch* is set as 50. In this experiment, the autoencoder can use the weight that is already learned. Then, the test time is very short. The training time is also short, because the autoencoder does not need to perform multiple layers of backpropagation.

TABLE 5
EXECUTION TIME (CLOSED-WORLD TEST)

Process	Description	Time [s]
Data Transmission	Time to convert train data and test data to Pylearn2 format.	124.4
Learning Time	Time to train using 7200 train data.	163.0
Test Time	Time to test 1800 Test data	3.0

5) Three-layer SDAE

The above results are obtained for the closed-world test by the two-layer SDAE. It is worthwhile investigating the performance of the three-layer SDAE.

We conduct another closed-world test using a three-layer SDAE. Table 6 shows the parameters of the new SDAE. The parameters of the MLP are the same as those in Table 3.

TABLE 6
PARAMETERS OF THREE-LAYER SDAE (CLOSED-WORLD TEST)

Parameter	1st Layer	2nd Layer	3rd Layer
nvis	5000	750	500
nhid	750	500	250
learning_rate	0.001	0.001	0.001
batch_size	50	50	50

Figure 7 shows the results while changing the learning intervals (*max_epochs*) every ten times. The maximum accuracy is 0.88, the same accuracy achieved by the two-layer SDAE. However, when three layers are used, the convergence of learning becomes slow compared with the two-layer SDAE.

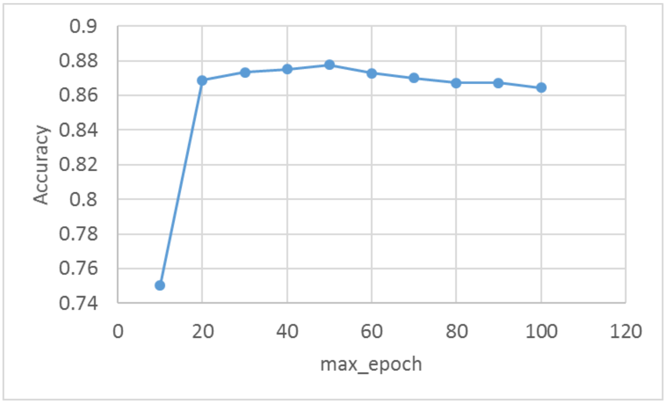

Fig. 7. Relation between max_epoch and accuracy in closed-world test.

The learning and test times also become slow. For the three-layer SDAE, when *max_epoch* is 50, the learning time becomes 241.9 s and the testing time becomes 3.6 s.

C. Open-World Test

1) Overview

In the open-world test, we use the data not only of 100 monitored sites, but also those of 9,000 non-monitored sites. The data of the monitored sites is divided into 72 instances for training data and 18 instances for testing data. We use 1,800 instances of non-monitored sites as the testing data.

A non-monitored website never appears in the training data. The victim can access a new website that the attacker does not expect. We label monitored websites as classes 0 to 99 and all the non-monitored websites as a single class 100.

2) Layer

In the open-world test, we use an MLP that has an input layer with a dimension of 5,000 and a two-layer DAE. The output layer is realized by a softmax function. The parameters of Pylearn2 are showed in Tables 7 and 8.

TABLE 7
PARAMETERS OF SDAE (OPEN-WORLD TEST)

Parameter	First Layer	Second Layer
Nvis	5000	500
Nhid	500	125
learrning_rate	0.001	0.001
batch_size	50	50
max_epoch	30	30

TABLE 8
PARAMETERS OF MLP (OPEN-WORLD TEST)

Parameter	MLP
Nvis	5000
n_classes	101
learning_rate	0.005
batch_size	200
max_epoch	50

3) Results

We investigate the TPR, i.e., the rate at which monitored websites are classified correctly, and the FPR, i.e.,., the rate at which a non-monitored website is classified as a monitored site. The TPR is shown in Figure 8, and the FPR is shown in Figure 9.

In Figure 8, when the number of training data instances of non-monitored sites is larger, the TPR is lower. The maximum TPR is 0.87, when the number of the training data of non-monitored sites is 1,000, and the minimum TPR is 0.86, when the number is 7,000. In Figure 9, in addition to the TPR, when the number of training data instances of non-monitored sites is larger, the FPR is lower. The minimum FPR is 0.02, when the number of training data of non-monitored sites is 7,000.

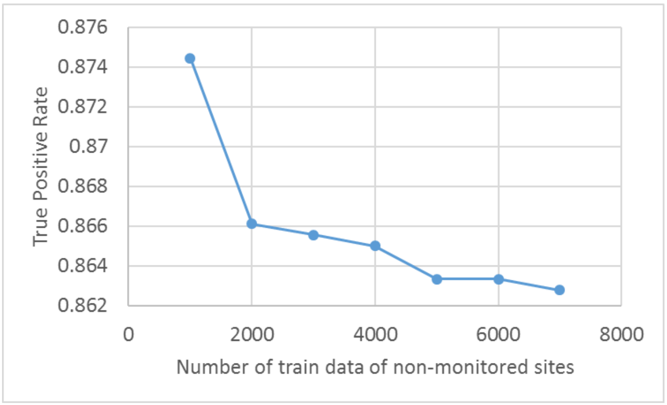

Fig. 8. Relation between number of training data of non-monitored sites and TPR in the open-world test.

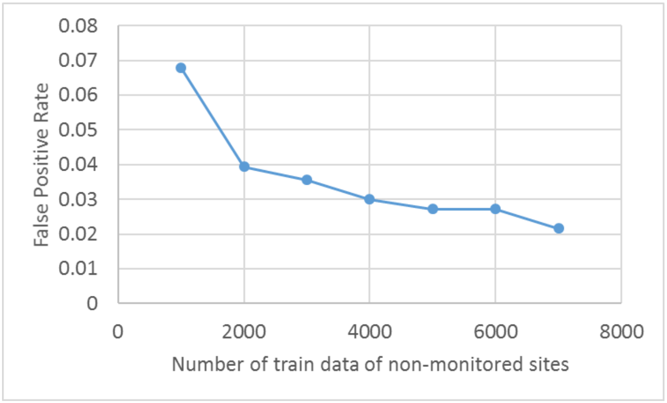

Fig. 9. Relation between number of training data of non-monitored sites and FPR in open-world test.

There is a trade-off between TPR and FPR. It is better to have a high value of TPR, while keeping the FPR value low.

Comparison with Related Work

Wang et al. showed the results of the OSAD [4] and k-NN methods [5] using the same dataset. Table 9 shows the comparison with previously known methods and our method.

In our proposed method, the accuracy in the closed-world test is 0.88, slightly lower than those of OSAD and k-NN. In the open-world test, our TPR (0.86) is higher than that of OSAD, and our FPR (0.02) is lower than that of OSAD. However, our FPR is higher than that of the k-NN method.

TABLE 9
COMPARISON WITH EXISTING METHODS

Method	Accuracy in Closed World Test	TPR in Open World Test	FPR in Open World Test
Our Method	0.88	0.86	0.02
OSAD Method	0.90	0.83	0.06
k-NN Method	0.91	0.85	0.006

VI. CONCLUSION

A. *Summary*

Here we propose a new method for fingerprinting attacks on Tor anonymity using SDAE. The input vector takes a very simple form, with elements 1, −1, or 0. The evaluation results show an accuracy of 0.88 in the closed-world test and TPR and FPR values of 0.86 and 0.02, respectively, in the open-world test. It is the advantage of our method that we can realize a high accuracy without selecting the features manually. Out method is based on mechanical Deep Learning.

This paper shows that deep-learning technology can be applied to fingerprinting attacks on Tor communications to have results comparable to those of existing technologies.

B. *Future Research*

It may be meaningful to combine our method with other methods proposed in related work. For example, the output of SDAE can be used as features in Wang's method and used for training by k-NN.

There are convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in deep learning. CNN has been used in pattern recognition. RNN can handle time-series data. It may be possible to improve the accuracy of our method by applying other neural network technologies as well.

ACKNOWLEDGEMENTS

A part of this work was supported by JSPS Grant-in-Aid for Scientific Research B, Grant Number 16H02832.

REFERENCES

[1] The Tor Project, "Tor Project: Anonymity Online," https://www.torproject.org/, referred Jan. 20, 2016.

[2] Roger Dingledine, Nick Mathewson and Paul Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, 2004, pp. 303–320.

[3] the LISA lab, "Welcome — Pylearn2 dev documentation," http://deeplearning.net/software/pylearn2/, referred Jan. 20, 2016.

[4] Tao Wang and Ian Goldber, "Improved Website Fingerprinting on Tor," WPES '13 Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society, 2013, pp. 201–212.

[5] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson and Ian Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23th USENIX Security Symposium*, 2014, pp. 143–157.

[6] Tao Wang, "Website Fingerprinting," https://cs.uwaterloo.ca/~t55wang/wf.html, referred Jan. 20, 2016.

[7] Alexa, "Alexa - Actionable Analytics for the Web," http://www.alexa.com/, referred Jan. 20, 2016.

[8] NVIDIA, "Parallel Programming and Computing Platform — CUDA — NVIDIA — NVIDIA," http://www.nvidia.com/object/cuda_home_new.html, referred Jan. 20, 2016.

[9] Andriy Panchenko, Lukas Niessen, Andreas Zinnen and Thomas Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *WPES '11 Proceedings of 27 the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 103–114.

[10] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson, "Touching from a distance: website fingerprinting attacks and defenses," *in CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[11] Hideki Asoh, Muneki Yasuda, Shiniti Maeda, Daisuke Okanohara, Takayuki Okatani, Yotaro Kubo and Danushka Bollegala, "Deep Learning," Kindai kagaku sha, Tokyo, 2015.

[12] Takayuki Okatani and Masaki Saito, "Deep Learning," IPSJ SIG-CVIM: Computer Vision and Image Media, 2013, pp.1–17.

Kota Abe Kota Abe received the B.S. degree in Computer Science and Engineering from Waseda University in March, 2016. He is now a master student at Department of Computer Science and Communications Engineering, Waseda University. His research interest covers Cyber Security.

Shigeki Goto Shigeki Goto is a professor at Depart-ment of Computer Science and Engineering, Waseda University, Japan. He received his B.S. and M.S. in Mathematics from the University of Tokyo. Prior to becoming a professor at Waseda University, he has worked for NTT for many years. He also earned a Ph.D in Information Engineering from the University of Tokyo. He is the president of JPNIC. He is a member of ACM and IEEE, and he was a trustee of Internet Society from 1994 to 1997.