Evaluation of Input Data Representations of Website Fingerprinting Attack on Deep Learning Performance

Mohammad Saidur Rahman

Abstract

Website fingerprinting (WF) is an attack by which an attacker can deanonymize a client of anonymity network such as Tor. A network level adversary who can observe network activates can perform this attack. The severity of this attack is so acute that making Tor secure in hardware level cannot ensure the user privacy [1]. Website fingerprinting attack is a machine learning classification problem. Previous research considered traditional machine learning (ML) techniques such as k-nearest neighbor (KNN), support vector machines (SVM), and random decision forests (RDF) for classification. Due to some basic limitations of those ML techniques such as manual feature selection, an attacker is inclined towards deep learning. Deep learning (DL) is a powerful way for feature selection. It can extract deep level features layer by layer. The major challenge of deep learning is to represent input data in the DL model. This project deals with this challenging issue. We experimented two input data representations. In both experiments, we are able to get more than 90% accuracy. We used stacked denoising autoencoder (SDAE) for our DL classification because of some obvious reasons such as extracting high-level features and dimensionality reduction of input data [2].

1 Introduction

With the emergence of the Internet, we are living in two worlds. One is the real world and the other one is the virtual world. The perception of connecting to people, gathering information, making ourselves

known to this universe is different in the virtual world than in real world. In the real world, we may not imagine knowing everybody and their personal information, but in the virtual world, it is possible. We are dependent on the Internet, and sometimes willingly or unwillingly we have to share our personal information over the Internet. This opens a door for privacy risk. At the same time, lots of efforts are being employed to enhance our privacy over the Internet.

Tor is one of the most popular privacy enhancing technologies. Tor enables users to browse the Internet anonymously. It also gives the freedom to communicate to the world where it is restricted or limited such as China. However, we cannot say that Tor ensures our complete privacy on the Internet. It also has some flaws that can be exploited by attackers to deanonymize users. An attacker can deanonymize a client by just observing network traffic. This attack is called website fingerprinting.

Website fingerprinting can be defined as the analysis of traffic to reveal which website a user is visiting. An attacker can deanonymize a user from that analysis using machine learning techniques such as k-nearest neighbor[3], support vector machine [4], and random decision forests [5]. Many research works have already been published proposing novel solutions to combat this attack [6-8]. But with the emergence of deep learning, an attacker can perform website fingerprinting attack with less effort and more efficiently. Only one research paper is published on website fingerprinting attack using deep learning that considered incoming and outgoing packets for input data [9]. But there are other ways to represent the input data. It will be efficient when an attacker can know which type of data representation she should feed to the model to train. This is one of the primary challenges of an attacker to perform website fingerprinting using deep learning.

This project deals with experimenting various ways to represent input data into the deep learning model. The objective is to find the best input data representation. Our default model is two-layer perceptron. And different activation functions will be used to see which one works best. Stacked denoising encoder will be used in designing deep neural network. We will take timestamp, outgoing packets, incoming packets and the combination of all to experiment as input format type.

The contribution of this project will help future research on website fingerprinting using deep learning, as researchers will have a clear idea about data representation. After feeding data, deep neural network can find pattern automatically without manual effort. If the data representation is erroneous, we can never get good prediction or output. This project is dealing with fixing this crucial problem of appropriate data representation.

2 Background

2.1 Tor

Tor is a low-latency anonymity network. A Tor circuit consists of three nodes (guard node, middle node, and exit nodes) (Figure 1). One node is only aware of the previous and next node of the circuit. Guard node only knows about the identity of the user and exit node only knows about the original TCP packets sent by the user [10]. Each node uses multilayered encryption. An eavesdropper, who can control the guard node and the exit node, can deanonymize the user.

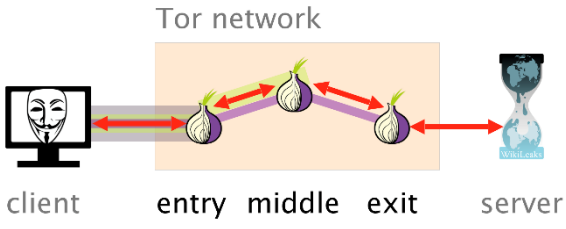

Figure 1: Tor Network [11].

2.2 Website Fingerprinting

We can achieve privacy by separating our activity from our identity. Our identity of

web browsing is our IP and our activities are the websites, we are visiting. Tor network ensures user privacy by keeping their identity anonymous.

Website fingerprinting is a type of attack that enables a local, passive eavesdropper to know the destination of a client analyzing traffic [12]. We assume that an attacker can monitor the traffic between a client and guard node, and/or she controls the guard node. She can collect statistical data about the traffic such as the total number of incoming packets, the total number of outgoing packets, the timing of each packet, and bursts. Packet burst is the sequence of incoming and outgoing packets. She can apply machine learning techniques to classify websites using those traffic features and use trained machine learning classifiers to identify the client's destination websites.

Website fingerprinting attacker collects traffic of several websites of her interest. These websites are called monitored websites. After that, she feeds these traffic data based on traffic features such as incoming and outgoing packets, bursts, and timestamps to machine learning model to learn from this input. This process is called training.

In the next step, she feeds her collected Tor traffic to the machine learning model to predict the new output. This process is called testing.

2.3 Neural Network

A neural network or an artificial neural network is a modeling tool. It is inspired by human neural systems. ANN imitates the learning process of human brain neurons. This concept was first introduced by Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts in 1943 [13]. They modeled a simple neural network with electrical circuits. An ANN estimate or approximate an unknown function that depends on inputs.

The neuron of an ANN takes inputs, and each input has the weight of their own (Figure 2).

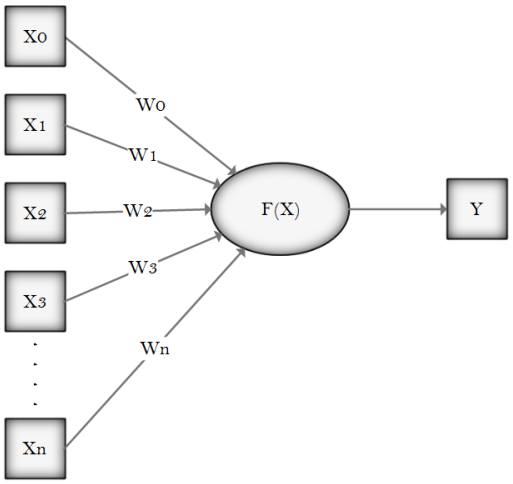

Figure 2: Simple Artificial Neural Network [14].

In Figure 2, F(X) is the neuron. A neuron consists of two functions: a summation function and an activation function.

The summation function works as follows:

$$X = \sum_{i=0}^{n} X_i W_i$$

The activation function can be of different types such as sigmoid activation function, tanh activation function, binary threshold neuron, rectified linear neuron, and the stochastic binary neuron. Each activation functions works in a different way. For instance, sigmoid activation function takes logit function to activate the function. It is calculated as follows:

$$Y = \frac{e^X}{1 + e^X}$$

2.5 Deep Learning

Deep neural network (DNN) is an artificial neural network that has multiple hidden layers. The complexity of a deep neural network is proportional to the number of hidden layers. It takes non-linear inputs and performs a complex computational task to predict the output.

Deep learning emerged to study the use of the DNNs for various applications. The structure and learning behavior of DNNs are different than general ANN, and other ML techniques. Specifically, Deep learning (DL) technologies are based on learning representations of data by means of effective algorithms for feature learning and feature extraction. One of the significant

advantages of deep learning (DL) is its ability to learn feature representations from unlabeled data.

For this project, stacked denoising autoencoders will be used for designing our experimental deep neural network (Figure 3). Autoencoding means to reduce the dimensions of the input data. Stacked autoencoder means stacking several autoencoders on each other and training them one by one.

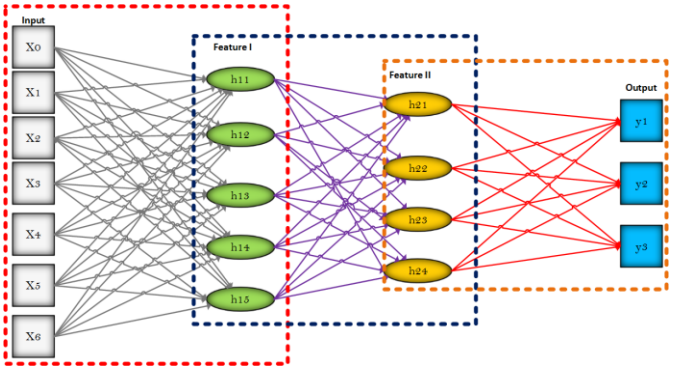

Figure 3: Stacked Encoder.

When we add noise to our input vector, the stacked autoencoder (SAE) becomes the stacked denoising autoencoder (SDAE). We add noise in our input vector in the following way (Figure 4).

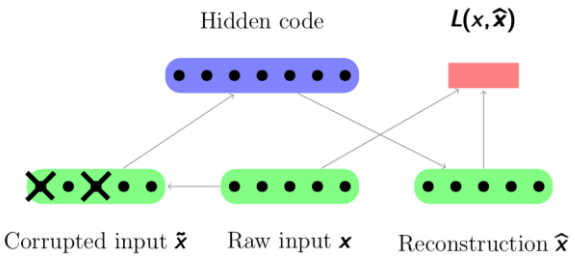

Figure 4: Process of Noise *Adding in Input Vector.*

After adding noise, the stacked autoencoder becomes stacked denoising autoencoder as follows (Figure 5):

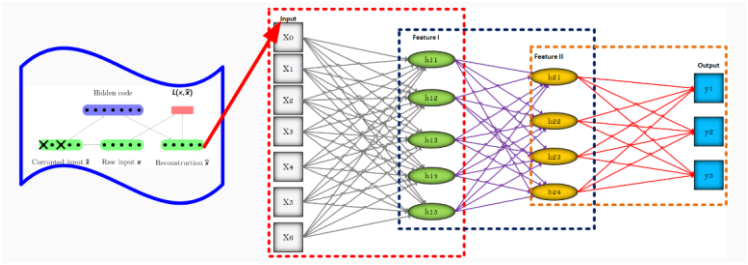

Figure 5: Stacked Denoising *Autoencoder.*

3 Related Work

3.1 Website Fingerprinting

In 2009, Herrmann et al. [15] developed a website fingerprinting attack in which they considered IP packet size for their classifier. Panchenko et al. designed a new attack adding more features: packet volume, packet direction, and timing [12]. They used support vector machines as their classifier.

SVM was again used by Cai et al. who proposed a new attack based on a new representation of the classification instances [16]. Their SVM was using the Damerau-Levenshtein edit distance and used the SVM kernel trick to pre-compute distance between the traces. This same attack was improved by Wang and Goldberg [17]. In 2014, Wang et al. proposed a new attack based on a k-Nearest Neighbor classifier on a large feature set with weight adjustment [3].

Hayes et al. use a novel feature extraction and selection method: they use random forests to extract robust fingerprints of webpages [18]. In 2016, Panchenko et al. proposed a new attack improving features based on packet size, packet ordering, and packet direction [19]. They used the concept of Wang et al. to develop their attack using k-nearest neighbor (K-NN) classifier.

3.2 Deep Learning

To the best our knowledge, only one paper has published so far that investigated the deep learning effectiveness on traffic analysis [9]. In this attack, their input data representation is based on incoming and outgoing packet traces. They got 88% of accuracy using deep learning without any manual selection of packet features.

However, they only considered one type of data representation omitting an important feature: timing of the packets. This project will experiment different input data representations to find out which one performs best.

4 Project Idea

The objective of this project is to find out the best input data representation to feed into the deep neural network for website fingerprinting attack. The initial plan of this project was to group the tasks into two broad categories as follows:

Step I

This project will try to evaluate three scenarios of data representation:

a. Both incoming and outgoing packets
b.
    i. Incoming packets only
    ii. Outgoing packets only
    iii. Concatenating I and II
c. Timestamp, incoming packets, and outgoing packets

Step II

a. Designing our deep neural network in SDAE mechanisms.
b. Experimenting with different activation function such as Sigmoid and Stochastic binary neuron (SBN)

The plan of the project work is shown in Figure 6.

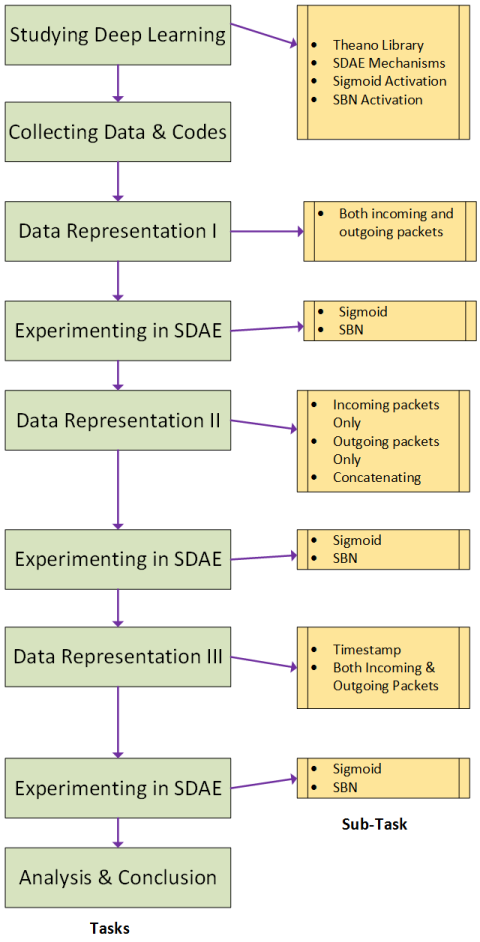

Figure 6: Plan of this Project.

However, at the time of our experiment, we had to modify our plan. Mainly, we completed our project considering two input data representations. We could not find any good way to represent timing of the packets. We tried to visualize the timing of the packets.

5 Experiment

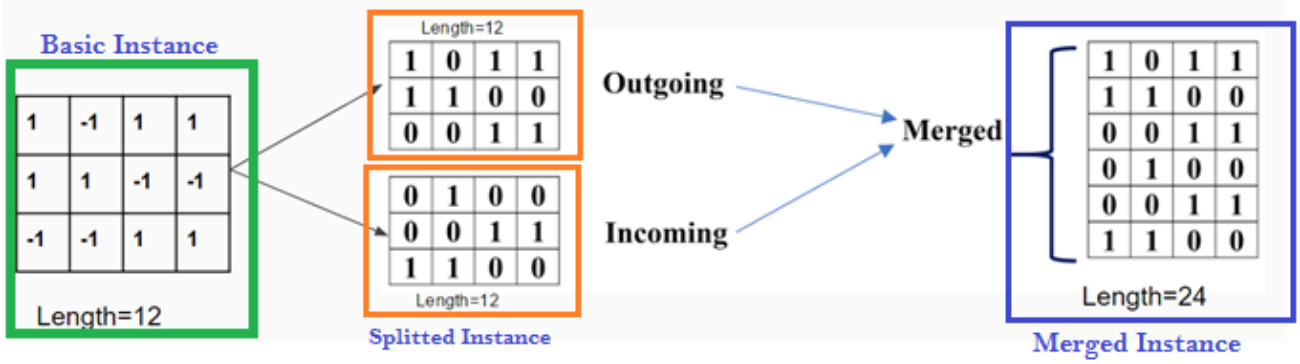

Figure 7: Concept of Input Data Representation.

Our major experiments are based on two input data representations (illustrated below).

Our both experiments are based on one major concept of input data representation, which is splitting outgoing and incoming packets and then combining these two into one (Figure 7).

In our basic instance, outgoing packets are denoted by +1 and incoming packets are denoted by -1. After that we halve the total length of the cell. In our spit instance, when we are considering outgoing packets, we put +1 for outgoing packets and we put 0 for incoming packets in that instance. When we are considering incoming packets, we put +1 for incoming packets and we put 0 for outgoing packets in that instance. Finally, we merge two instance together.

5.1 Experiment I

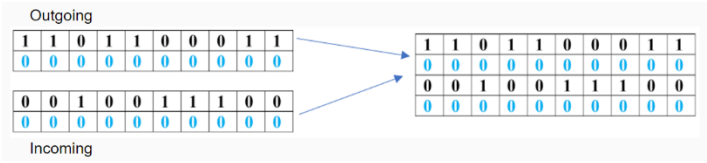

Figure 8: Appending Zero.

In our first experiment, we compare actual number of lines in a cell. If it is greater than or equal to the length, we put 1.0 for outgoing packets and 1.0 for incoming packets in our split instance. If the actual number of lines in a cell is less than the

length, we put 1.0 for outgoing packets and 1.0 for incoming packets in our split instance for the lines exist in the cell. We pad the rest of the lines by 0.

Experiment in this input data representation was conducted with two, three and four hidden layers. We also saw the impact of changing the length of the cell. We also observed the impact of filtered length.

5.2 Experiment II

This experiment is similar to the experiment one with an exception. Instead of padding empty lines with 0, we copied the outgoing and incoming split instance to fill out the length of the cell.

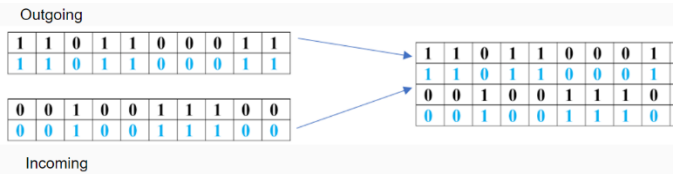

Figure 9: Copied Instance.

5.3 Experiment with Timing

Initially, we could not figure out any way to represent timing of packets direction. After that, we decided to find if any pattern in timing of packets direction exist. We tried to see some graphs of each websites and compare them. In this experiment, we took ten instances of each website. We took six websites for this experiment.

5.4 Experimental Setting

We used Linux (Ubuntu) environment for our experiment. We used GPU for faster computation. We used Theano deep learning library. Stacked denoising autoencoder is our deep learning classifier. We used the data of Wang et al. [3]. We used the codes of Theano tutorial [20] and modified the codes in accordance to our requirements.

6 Results

6.1 Result of Experiment I

Config. No.	Input Layer Length	Hidden Layer		Input Instance			Filter	Accurac (%)
		1	2	Training	Validation	Testing		
1	5000	1500	1000	58	14	18	200	84.52
2	5000	1500	1000	72	1	17	200	87.97
3	5000	1500	1000	81	1	8	200	**89.97**

Figure 10: First Phase of the Experiment.

The default input layer length is 5000, the number of training, validation and testing input instances are 58, 14 and 18, and the number of length for filtering packets is 200. In the default configuration, the accuracy is 84.52%. Each of the experiment was run five times and took the average of the accuracy. We can see from Figure 10, that increasing the number of training instances gives us better accuracy.

In two hidden layers, we got at best 91.72% accuracy with 4000 input layer length; and with 72 training, 1 validation, and 17 testing instances; and the length of the packets to be filtered is 500 (Figure 11).

Config. No.	Input Layer Length	Hidden Layer		Input Instance			Filter	Accuracy (%)
		1	2	Training	Validation	Testing		
10	4000	1500	1000	58	14	18	500	87.05
11	4000	1500	1000	72	1	17	500	**91.72**
12	4000	1500	1000	81	1	8	500	91.22

Figure 11: Best Accuracy in Two Hidden Layers.

However, we got the best accuracy in this input data representation with three hidden layers (Figure 12).

Config. No.	Input Layer Length	Hidden Layer			Input Instance			Filter	Accuracy (%)
		1	2	3	Training	Validation	Testing		
27	5000	2000	1500	500	72	1	17	500	90.18
29	**4000**	**2000**	**1500**	**500**	**72**	**1**	**17**	**500**	**91.89**

Figure 12: Best Accuracy *in Three Hidden Layers.*

6.2 Result of Experiment II

In our second experiment, we got the best accuracy 90.77 with four hidden layers; and 75 training, 1 validation, and 14 testing instances; and the length of the packets to be filtered is 500 (Figure 13).

Config. No.	Input Layer Length	Hidden Layer				Input Instance			Filter	Accuracy (%)
		1	2	3	4	Training	Validation	Testing		
6	4000	1500	1000	750	500	58	14	18	500	86.23
7	4000	1500	1000	750	500	65	5	20	500	88.23
8	4000	1500	1000	750	500	70	5	15	500	88.86
9	4000	1500	1000	750	500	75	1	14	500	90.34
10	3600	1500	1000	750	500	75	1	14	500	**90.77**

Figure 13: Best Accuracy in Four Hidden Layers.

6.3 Result of Timing Experiment

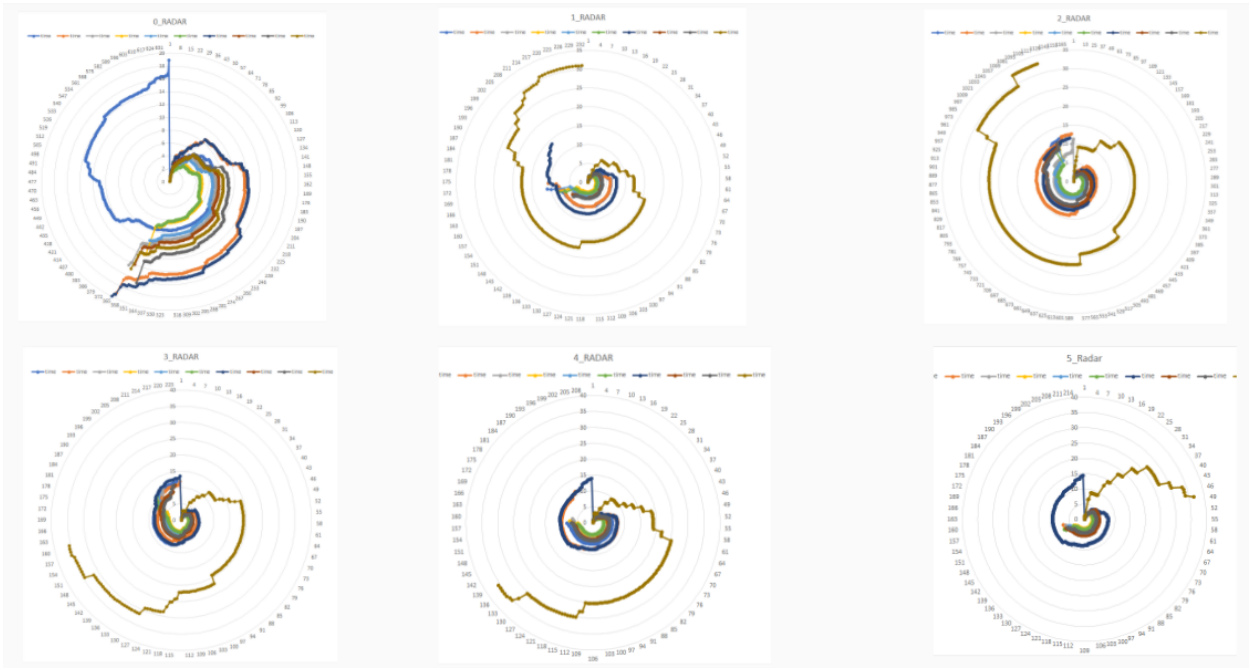

Figure 14: Visualization in Radar Graph.

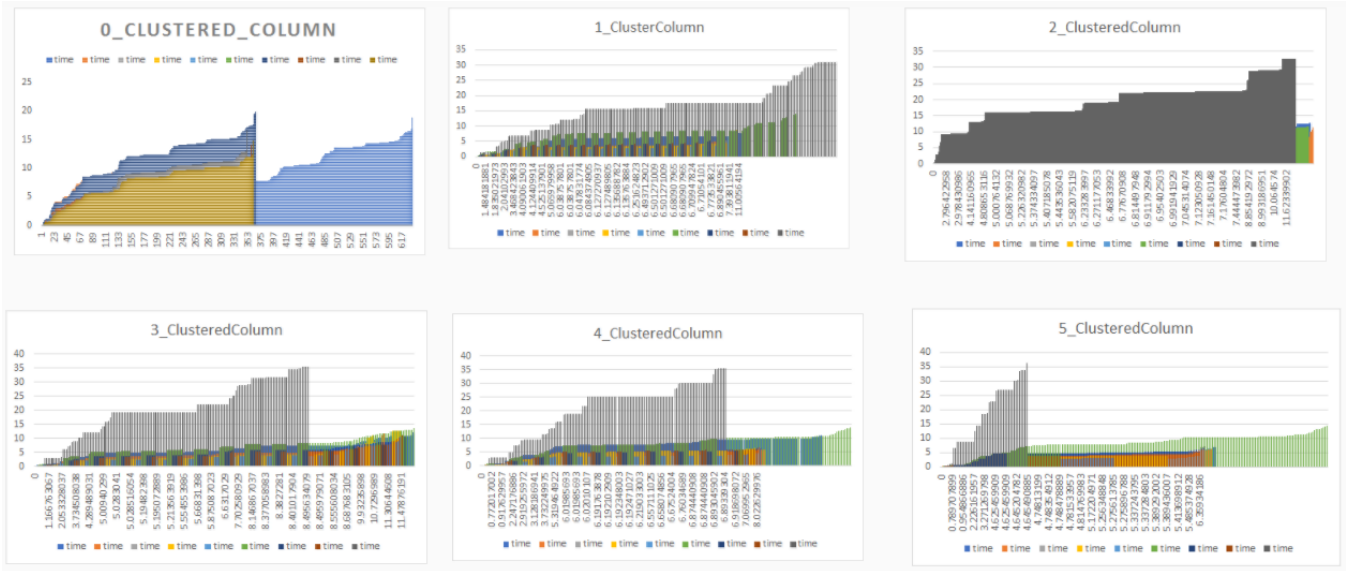

Figure 15: Visualization in Clustered *Column Graph.*

6.4 Result Analysis

From all our experiment, we came to know several important issues. Firstly, the length of input layer is important. We cannot fix the length of each hidden layer units arbitrary. We can see from our experiment, when we maintain some proportion, the deep learning classifier gives us better accuracy. We also observed that the number of training instances matters a lot. This is obvious because we know that deep learning requires lot of data for training. In our experiment with timing of packets direction, we could not find any pattern. Even we tried some visualization (Figure 14 and Figure 15), but we could not see any pattern. However, we can see that, the best input data representation is representation of experiment one. We plan to use this representation in our future work.

7 Limitations and Future Work

We think the major limitation of this project is that we had to change the number of training, validation and testing instances from standard ratio. We got the best accuracy in 72 training instances, 1 validation instance, and 17 testing instances. One validation instance does not give us best validation. It is not even standard in machine learning. We did not have any error checking such as cross-validation.

We are going to work in future to overcome this limitation increasing the number of data. We have already collected big data set for our new experiments. We used Theano library for our deep learning model. We are planning to use Pylearn library because Abe & Goto [9] used this library. As we want to compare our results with them, we need to use the same library they used. We will go in

depth literature to find if there is any proportional relationship between the lengths of each two layers such as input layer to first hidden layer, and first hidden layer to second hidden layer, and so on. We will try to experiment timing of packets direction in recurrent neural network (RNN) because RNN considers timing exclusively. We will also experiment our work in convolutional neural network. We will experiment the effectiveness of our attack against the defenses of website fingerprinting.

8 Conclusion

In this project, we tried to experiment two input data representations. Initially we planned use timing of packets direction in our experiment, but we could not find any way for that. We introduced a new concept of input data representation that is splitting the basic instance into two instances and then merging. We experimented this concept in two ways. One is padding with zero (Figure 8) and another is copying the instances (Figure 9). We tried different hidden layers in our experiment. We got at best 91.89% accuracy in first input data representation (experiment one) with three hidden layers.

9 Acknowledgement

Firstly, we are grateful to Professor Dr. Matthew Wright for continuous support and feedback to develop this project. After that, we would like to express our gratitude to Marc Juarez and Vera Rimmer for rendering their codes and data. After that, I would like to thank Mohsen Imani for his support in collecting data set. Finally, I would like to express my special gratitude to Payap Sirinam for his continuous support to develop concepts of this project, conduct the experiments, and providing feedbacks.

References

[1] S. Kim, J. Han, J. Ha, T. Kim, and D. Han, "Enhancing Security and Privacy of Tor {\textquoteright} s Ecosystem by Using Trusted Execution Environments," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 145-161: USENIX Association.

[2] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research,* vol. 11, no. Dec, pp. 3371-3408, 2010.

[3] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses forWebsite Fingerprinting," San Diego, CA, USA, 2014: 23rd USENIX Security Symposium.

[4] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor," Berlin, Germany 2013, pp. 201-212 Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society

[5] J. Hayes and G. Danezis, "k-fingerprinting: a Robust ScalableWebsite Fingerprinting Technique," Austin, TX, USA, 2016: 25th USENIX Security Symposium

[6] T. Wang and I. Goldberg, "Walkie-Talkie: An Effective and Efficient Defense against Website Fingerprinting," 2015.

[7] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*, 2016, pp. 27-46: Springer.

[8] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 227-238: ACM.

[9] 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.

[10] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document2004.

[11] T. P. E. Lab. (2015). *The Tor Network*. Available: http://scenic.princeton.edu/pt-tor-network.php

[12] 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*, 2011, pp. 103-114: ACM.

[13] I. Russell. (1996). *Neural Networks Module*. Available: http://uhaweb.hartford.edu/compsci/neural-networks-history.html

[14] S. Barber. (2007). *AI : Neural Network for beginners*. Available: https://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of

[15] 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, 2009, pp. 31-42: ACM.

[16] 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*, 2012, pp. 605-616: ACM.

[17] 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*, 2013, pp. 201-212: ACM.

[18] J. Hayes and G. Danezis, "Website fingerprinting at scale," *arXiv preprint arXiv:1509.00789,* 2015.

[19] A. Panchenko *et al.*, "Website fingerprinting at internet scale," in *Network & Distributed System Security Symposium (NDSS). IEEE Computer Society*, 2016.

[20] U. o. M. LISA lab, *Deep Learning Tutorial*. 2015, p. 173.

Appendix

All the results of experiment I

Each configuration is experimented five times.

Config. No.	Input Layer Length	Hidden Layer				Input Instance			Filter	Accuracy (%)
		1	2	3	4	Training	Validation	Testing		
1	5000	1500	1000			58	14	18	200	84.52
2	5000	1500	1000			72	1	17	200	87.97
3	5000	1500	1000			81	1	8	200	89.97
4	4000	1500	1000			58	14	18	200	86.41
5	4000	1500	1000			72	1	17	200	90.19
6	4000	1500	1000			81	1	8	200	91.03
7	3000	1500	1000			58	14	18	200	84.93
8	3000	1500	1000			72	1	17	200	89.26
9	3000	1500	1000			81	1	8	200	89.81
10	4000	1500	1000			58	14	18	500	87.05
11	4000	1500	1000			72	1	17	500	91.72
12	4000	1500	1000			81	1	8	500	91.22
13	5000	1500	1000			58	14	18	500	85.98
14	5000	1500	1000			72	1	17	500	90.10
15	5000	1500	1000			81	1	8	500	90.01
16	5000	2000	1000			72	1	17	500	85.98
17	5000	1500	750			72	1	17	500	86.06
18	4000	2000	1000			72	1	17	500	86.67
19	4000	1500	750			72	1	17	500	86.60

20	5000	2000	1500	1000		58	14	18	200	85.67
21	5000	2000	1500	1000		72	1	17	200	88.97
22	5000	2000	1500	1000		58	14	18	500	86.29
23	5000	2000	1500	1000		72	1	17	500	90.43
24	5000	1500	750	500		72	1	17	500	89.69
25	5000	1500	1000	500		72	1	17	500	89.53
26	5000	2500	1500	500		72	1	17	500	90.51
27	5000	2000	1500	500		72	1	17	500	90.18
28	4000	2500	1500	500		72	1	17	500	91.48
29	4000	2000	1500	500		72	1	17	500	91.89
30	4000	1500	750	500		72	1	17	500	90.79
31	5000	2500	1500	1000	500	72	1	17	500	90.31
32	5000	2000	1500	1000	500	72	1	17	500	90.48
33	5000	1500	1000	750	500	72	1	17	500	88.81
34	4000	2500	1500	1000	500	72	1	17	500	91.67
35	4000	2000	1500	1000	500	72	1	17	500	91.56
36	4000	1500	1000	750	500	72	1	17	500	91.18

All the results of experiment II

Each configuration is experimented five times.

Config. No.	Input Layer Length	Hidden Layer				Input Instance			Filter	Accuracy (%)
		1	2	3	4	Training	Validation	Testing		
1	5000	1500	1000	500	250	58	14	18	200	80.06
2	5000	1500	1000	500	250	58	14	18	50	65.6
3	3000	1500	1000	500	250	58	14	18	250	80.80
4	3000	1250	1000	750	600	58	14	18	500	84.33
5	4000	1500	750	500	350	58	14	18	500	85.13
6	4000	1500	1000	750	500	58	14	18	500	86.23
7	4000	1500	1000	750	500	65	5	20	500	88.23
8	4000	1500	1000	750	500	70	5	15	500	88.86
9	4000	1500	1000	750	500	75	1	14	500	90.34
10	3600	1500	1000	750	500	75	1	14	500	**90.77**