All for One: Improving a Fingerprinting Attack on Tor Using Ensemble Learning

Dhruv Kandula
Cupertino High School
Cupertino, California

***Abstract*—Tor is a free software that provides anonymity and privacy for its users. This system is especially useful for people like whistle blowers and journalists who live in countries where free speech is prohibited, as it allows them to make actions without fear of being caught as it is done anonymously. However, there is an attack called "website fingerprinting" which can identify which website a user on Tor accesses. This website fingerprinting attack can be conducted through the use of machine learning models. An example is an SDAE(Stacked Denoising Auto-Encoder) deep learning model that could identify what website a user accesses simply by observing the user's interactions with the Tor network. This paper uses a method called ensemble learning which can be used to improve previous deep learning website fingerprinting attacks like the SDAE. By simply increasing the number of SDAE models, the collective can achieve much greater results than the single SDAE in benchmarks such as precision, recall, and the F1 score.**

***Index Terms*—Tor, website fingerprinting, deep learning**

I. Introduction

The Onion Routing network, known commonly as Tor, is a free and open-source software that allows people to browse the internet anonymously [1]. This is possible thanks to an international network of relays run by volunteers. These relays help people protect their privacy by hiding the user and the website the user is on. At most, any entity that spies on the relays can either know who is using Tor, or that someone is on X website. They can only know one or the other, not both. The Tor network is especially useful for people who live in countries where their freedom of speech is inhibited, as it allows them to say anything anonymously without fear of being caught.

Tor is important for everyone. It can help people preserve their privacy by not letting third party trackers constantly monitor their every action on the internet. It can also help people access websites that their government may have banned, such as WikiLeaks and other sites that provide information that the government wouldn't want people to know. While Tor may be known for a fraction of the users that participate in illegal activities, it is also used by many ordinary people and people whose online privacy is vital. This could include political activists, journalists, anyone living in oppressive regimes, etc. Tor lets users avoid censorship, spying, and profiling from 3rd party organizations.

There is an attack called website fingerprinting that can identify what website a user is accessing using their communications with the Tor network. This website fingerprinting attack can be conducted through the use of machine learning algorithms, and more specifically, deep learning neural networks. One previous deep learning neural network was the Stacked Denoising Auto-Encoder, known as the SDAE [2], which can analyze captured encrypted packets, identify the website a user accesses simply by examining the directions of these packets. This paper uses a method, ensemble learning, which can be used to enhance previous deep learning attacks like the SDAE. By simply increasing the number of models, even if the models are worse than the original SDAE, the collection of models can surpass the original model in precision by 3.5%, recall by 4.6%, and the F1 score by 4.8%.

II. Background

A. Tor

Tor works by sending a user's message on the internet through a connection of 3 proxies. Any message that a user sends will go through a guard node, which sends the message to a relay node, which then sends the message to an exit node, which finally sends the message to the target destination. This ensures privacy and anonymity on the internet by not letting the target web server know what your IP address is, since it only sees the IP address of the exit node in the Tor system.

As seen in Figure 1, Whenever a message is sent to a web server, it is encrypted 3 times. Encryption is the process of taking some data and turning it into a scrambled code. Anyone who sees this scrambled code will not know what the true message is, but people with the correct key will able to decode it into the real message. This ensures confidentiality, so only authorized people with the correct key can read the message. Each Tor node will have a single key to take away one layer of encryption. Once the message goes through the final exit node, the message won't have any more Tor encryption, but it will most likely still be encrypted as long as the target website uses the HTTPS protocol. When a message is sent from the target server back to the user, the opposite happens. The message will be encrypted once by each Tor node on the way to the user. When it reaches the user from the original entry node, the message will have three layers of encryption. Unlike the Tor nodes, the user has all three keys instead of one, so the user can decrypt the entire message and see its contents. The reason Tor is able to protect privacy so well is because of the three relays, or nodes. The first relay, known as the guard node will know who is using Tor and which middle node to send the message to, but nothing else. It will not know the true

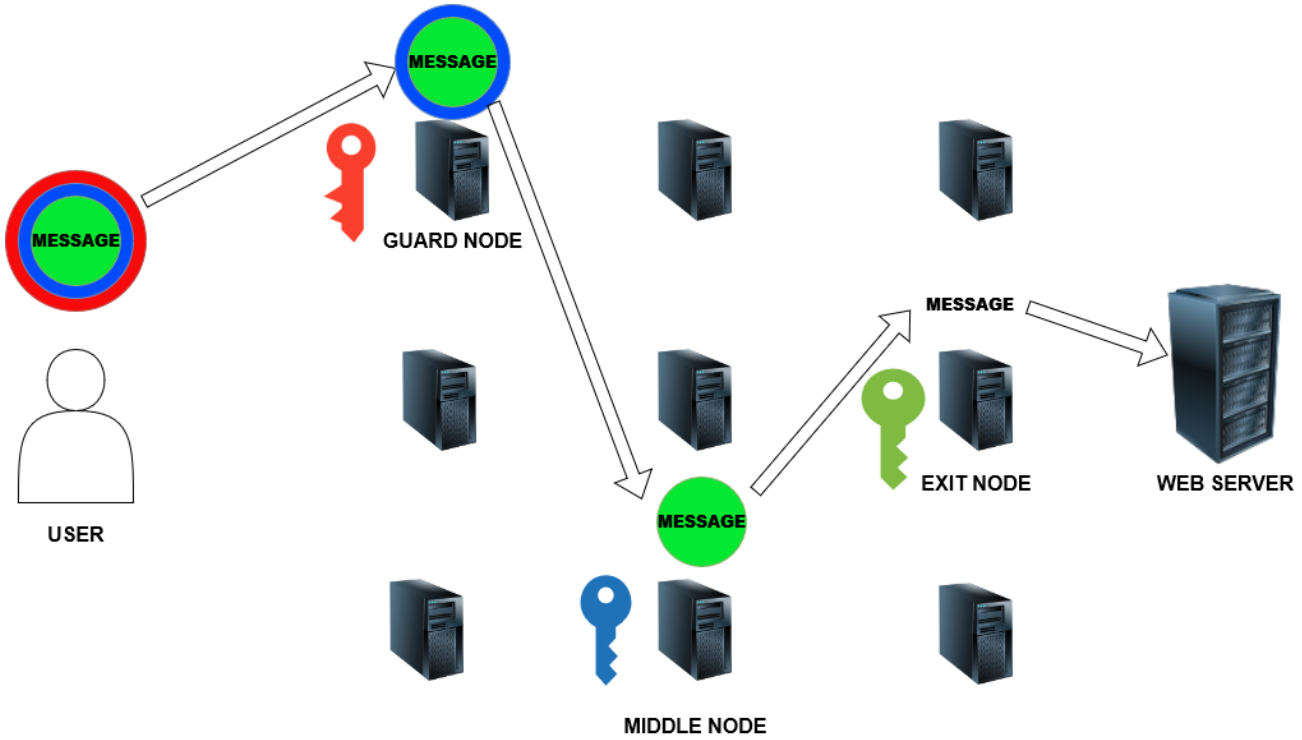

Fig. 1. Tor diagram.

contents of the message because there will still be two layers of encryption, and it won't know the real destination of the message. The middle node will only know which guard node sent it the message, and which exit node to send the message to. It won't know who is using Tor, the destination of the message, or the actual message because there will still be one more layer of Tor encryption. The final node, known as the exit node, will only know the destination of the message, the middle node which sent it the message, and also the content of the message if the website doesn't use the HTTPS protocol. Since it has the final key, there will be no more layers of Tor encryption. However, the exit node will not know the user who is accessing the target website. If an attacker somehow compromises one of the nodes, they won't be able to see the whole picture. For example, if a malicious entity had control of the guard node, they would only know who is using Tor, but not the website that they are on or what messages they are sending, since it is encrypted. Similarly, if an attacker has control of the exit node they would only know that someone sent some message to the web server, but they won't know who did it.

B. Website Fingerprinting Attacks

A website fingerprinting attack is an attack that aims to identify what website a person is using, even if that person is using Tor. This is achieved by examining the packets that go from the user to the guard node and the packets that go from the guard node to the user. This is possible if an attacker is able to perform a "man in the middle" attack where they can see the packets of information going to and from the user. This could be conducted by an ISP provider or another type of network operator [2]. It could also be performed if a malicious threat took control of a guard node [2]. This second scenario is very unrealistic because Tor guard nodes are are generally trusted. Even if they were compromised they wouldn't be able to target specific individuals because Tor assigns guard nodes at random, and only rotates the chosen guard nodes every 2-3 months [3].

C. Deep Learning

Deep learning is a type of machine learning that leverages the use of neural networks to accomplish its task [4]. A machine learning algorithm takes in input in the form of numbers, and gives an output in the form of numbers. The algorithm learns from a lot of data, where each piece of data has some input numbers and some output numbers, and it finds some correlation between the input and output. A typical neural network has three sections: an input layer, some hidden layers, and an output layer. A very basic neural network is shown in Figure 2. Input data in the form of numbers will be fed into the input layer of the neural network. The size of the input will be equal to the size of the input layer. For example, in Figure 2, the input layer has 5 'neurons', so it can take an input with 5 numbers. Each circle in Figure 2 is known as a 'neuron'. The line connecting the neurons together is known as a weight. The weight is a number, which will be multiplied by the number the neuron holds. In this example, each neuron in the hidden layer has 5 lines connecting to it. Each line is

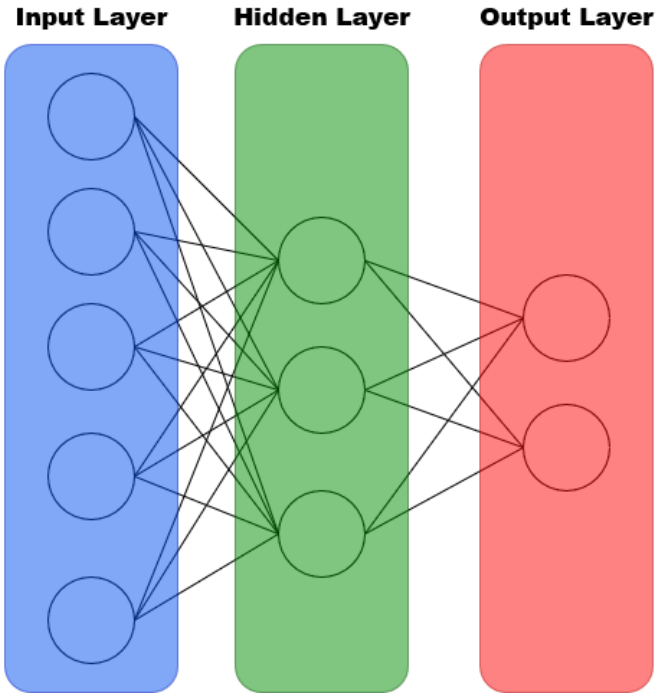

Fig. 2. Basic neural network diagram.

a different weight that will be multiplied with the number the neuron that is connected to it holds. Then, the outputs of all the weights will be added up, so in this case each hidden layer will add up 5 numbers. Then, this sum will go through an activation function, and the output of that function will be the value that the neuron will hold. Finally, the output layer will generate a list of numbers, and the length of this list will be equal to the number of neurons in the output layer. The output will then be compared to the actual answer of the input, and based on this error, the weights will be changed. This process will repeat until the weights reach the ideal value so that the output will produce a minimal error. This process of changing the weights based on the error is called "back propagation". Each layer is defined by its input size, the number of lines connecting to each neuron, and the output size, the number of neurons in a layer. In Figure 2, the hidden layer has an input size of 5, and and output size of 3.

D. metrics

In order to analyze a deep learning model, we can use many different metrics. In our paper, we will be examining the precision, recall, and the F1 score of the model.

The "positive" class is when a model is saying that "this is X", while negative is when the model says "this is not X". True positive is when the model correctly identifies the positive class. If something is X, and the model says "this is X", then it is true positive. A false positive, on the other hand is when the model incorrectly predicts a positive class. If something is Y, but the model predicts that it is X, then it is a false positive. A true negative is when the model predicts a negative class correctly. If something is Y, and the model says "this is not X", it is a true negative. Finally, a false negative is when the model incorrectly predicts a negative class. If something is X, but the model says "this is not X", it is a false negative.

The precision of a deep learning model is the True positive divided by the sum of true positive and false positive. This metric shows the number of times the model correctly guessed the correct web page divided by the total number of times it guessed that web page.

The recall of a model is the True positive divided by the sum of the True positive and the False negative. The recall is the number of times the model correctly guessed the correct web page divided by the total number of times the web page appears in the dataset.

The F1 score of the model is the product of precision and recall divided by the sum of precision and recall. This metric combines the precision and recall together.

E. SDAE

The SDAE, or Stacked Denoising Auto-Encoder, is a deep learning architecture [2]. The Auto-Encoder is a deep learning model which has three parts: input layer, hidden layer, and output layer. The unique aspect of an Auto-Encoder is that the output layer and input layer have the same size. A Stacked Auto-Encoder is where instead of a hidden layer, there is another Auto-Encoder between the input and output layer. The Auto-Encoder can be stacked indefinitely, but in our paper we only use one stack, so there is only one Auto-Encoder inside of the main Auto-Encoder. A Stacked Denoising Auto-Encoder is a Stacked Auto-Encoder with noise in the input, so the SDAE will have to remove the noise, and in doing so gain a better understanding of the data.

III. Related Work

A. Fingerprinting Attack

In 2013, Goldberg and Wang [5] conducted a website fingerprinting attack on Tor. They collected Tor data on 100 websites, with 40 instances each. They then fed this data into a machine learning algorithm called an Support vector Machine, which would be able to predict which website the user was accessing.

In 2014, Wang et al. [6], used another machine learning algorithm called the K Nearest Neighbors(KNN) algorithm. This time they collected Tor data over 100 websites, with 90 instances each. From the Tor packet data they monitored, they extracted a multitude of features and fed them to the machine learning algorithm to perform the website fingerprinting attack. These features included: total transmission size, total transmission time, numbers of packets coming to and from the user, unique packet lengths, packet ordering, lengths and direction of the first 20 packets, and concentration of outgoing packets. They also use bursts, which they define as "a series of outgoing packets, in which there are no two adjacent incoming packets". They extract features from bursts as well, such as the mean burst length, maximum burst length, and number of bursts. In total they used approximately 4000 features. They then fed

these features into the KNN machine learning algorithm to conduct a website fingerprinting attack.

B. Attacks using Deep Learning

In 2016, Abe and Goto [2] made a deep learning model that utilized the dataset used in the paper by Wang et al. [6] They used what is known as a Stacked Denoising Auto encoder, otherwise known as SDAE, to conduct this attack. While the paper by Wang et al. [6] had to extract approximately 4000 features to input into a machine learning model, the paper by Abe and Goto [2] only used the cell's directions as the input. In this paper, we will improve upon this attack using an ensemble of SDAE models.

In 2018, Bhat, Lu, Kwon and Devadas [7], created a state of the art deep learning website fingerprinting attack. Instead of using the SDAE like Abe and Goto [2], they used a different type of deep learning model called a Convolutional Neural Network (CNN). They use a dataset of 900 websites, each with 2500 instances, which is much more data than the paper by Wang et al. [6] used.

Our work is different, because while the previous papers listed use new deep learning models, we use a method of improving models instead of providing a new one. In our paper, we will be improving the SDAE model used by Abe and Goto [2] not by proposing a new type of model, but by increasing the number of SDAE models. This method can be implemented in other models also to increase their benchmark scores.

C. Mitigation against Fingerprinting Attacks

In 2018, Wang and Goldberg [8] constructed a defense against website fingerprinting attacks called Walkie-Talkie. The Walkie-Talkie defense makes any web browser communicate with the user in half-duplex mode instead of full duplex mode. In full duplex mode, the user and website can communicate with each other simultaneously, like when two people talk to each other via telephone. However, in half duplex mode, only one side can communicate with the other at a time, like when two people talk over a walkie-talkie. This defense can mold burst sequences to leak less information to malicious entities, combating website fingerprinting attacks effectively and efficiently.

IV. Methodology

This paper will use the same dataset that was used in Wang et al. [6] and in Abe and Goto [2]. This dataset is publicly available in https://github.com/kdsec/wangknn-dataset, which mirrors its data from https://www.cse.ust.hk/ taow/wf/data/. This dataset consists of 100 monitored websites, which are numbered from 0 to 99. Each monitored website has 90 cell traces, numbered 0-89. $100 \times 90 = 9000$, so the dataset has exactly 9000 cell traces. Each trace contains a list of cells going to from the user to the Tor guard node, and from the guard node back to the user. This list has two columns: the first column is the timestamp of when that cell is captured, and the direction of the cell. The timestamp of each cell is relative to the first cell sent. The first cell will always have a timestamp of zero, and all proceeding timestamps will be the number of seconds that a cell was caught after the first one was sent. If a timestamp says 5.0, this means that this cell was sent 5 seconds after the first cell is sent. Since Abe and Goto [2]'s deep learning attack did not make use of the timestamps, we can disregard this column and only use the second column. The second column describes the direction of the cell in the form of a 1 or -1. If it is a 1, then the cell has been sent by the user to the Tor guard node. If it is a -1, then the cell has been sent from the guard node to the user. In this attack, we will only be using the directions of the cells as input to identify which website is being used.

TABLE I
EXAMPLE OF A TRACE.

Timestamp	Direction
0.0	1
0.111994028091	1
0.350380897522	-1
0.465469121933	-1
0.524447917938	-1
0.524447917938	-1
0.524447917938	-1
0.524447917938	-1

Table I displays the first 8 cells from the file 0-33, which is the 34th trace of the first monitored website.

In order to feed the data into a deep learning model, we will turn the directions of the cells for each trace into a single vector of a fixed size. All inputs for our deep learning model needs to be the same size, but the problem with this dataset is that each trace has a different size, as seen in Figure 3.

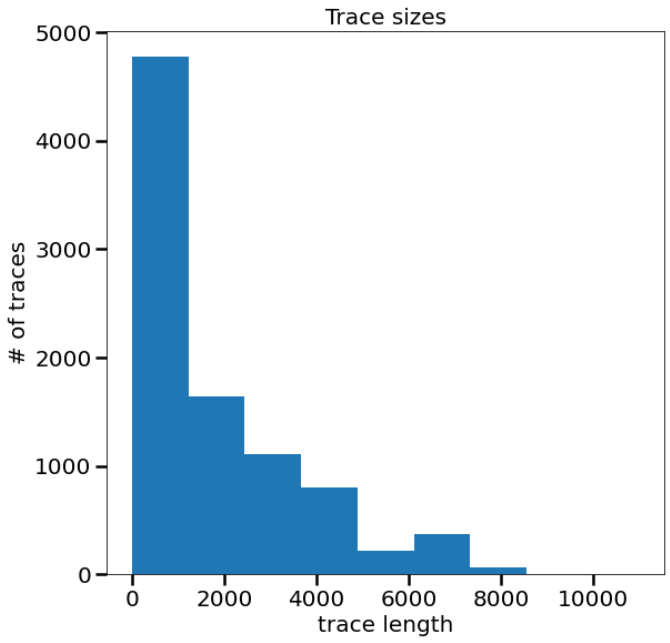

Fig. 3. A plot of the number of traces for each approximate trace length.

Abe and Goto [2] solve this problem by forcing each vector

to be of length 5000. If the vector length is more than 5000, it gets truncated. If it is less than 5000, the vector gets padded with zeros. In the end, the input will be a vector of length 5000 with values that are either 1, -1, or 0.

A. One-Hot Encoding

Before we train our deep learning model on the input vectors, we must first one-hot encode our expected output data. Out deep learning model will be a **classification** model. This means that our model will use the input data to classify which website the trace refers to out of the hundred monitored web pages. In order to make a classification model, the output data must be one-hot encoded.

One-hot encoding is the process of converting a number or label into a vector consisting of ones and zeroes. The length of the vector will be equal to the number of unique classes in the data. In this case, there are 100 classes because there are 100 monitored web pages, so the length of the vector will be 100. For each label, there will be 99 zeroes and a 1 at the index of the web page number. For example, the first monitored web page, web page no. 0, will be represented as: [1, 0, 0, 0, 0... 0]. The second web page, web page no. 1 will be shown as: [0,1,0,0,0,0,... 0].

B. Preparing Dataset

Before training the model, the dataset must be randomized. The, the data is split up into a train dataset and a test dataset. This is to ensure that the deep learning model does not **overfit** on the data. If the model overfits on the data, then it will simply memorize the inputs and their corresponding outputs instead of actually learning their relationship with each other. The train data will be 80% of the entire data, and the test data will be 20% of the whole data. Since each web page has 90 traces, 18 of the traces will be part of the test dataset, and the other 72 traces of each web page will be part of the train dataset. When splitting up the data, we need to ensure that the number of traces from each web page is the same for both datasets so that the model can properly train on the input data.

After it is split, the data in each dataset is spread out into batches of size 50, as done in Abe and Goto [2]. It was randomized earlier so that each batch wouldn't have similar trace inputs. It is batched in order to save memory. Instead of training on all 9000 traces all at once, the model will train on batches of 50 traces at a time.

C. SDAE Model

The deep learning attack that we will improve upon is the 2 layer Stacked Denoising Auto-Encoder model used in the paper by Abe and Goto [2].

The architecture for this model is shown in Figure 4. This Stacked Auto-Encoder is symmetric, in the sense that the first layer will have the same input size as the last layer's output size. The second layer's input size will be the same as the output size as the penultimate layer's input size, and so on. For this specific model, the input size of 5000 and the output size of 100 can't be changed during experimentation in order for the model to use the dataset. In the paper by Goto and Abe [2], the SDAE model with the best results had the second layer have an output layer of size 750, and the center layer have an output of size 250. Finally, the output layer has a softmax activation with an output size of 100, because there are 100 classes.

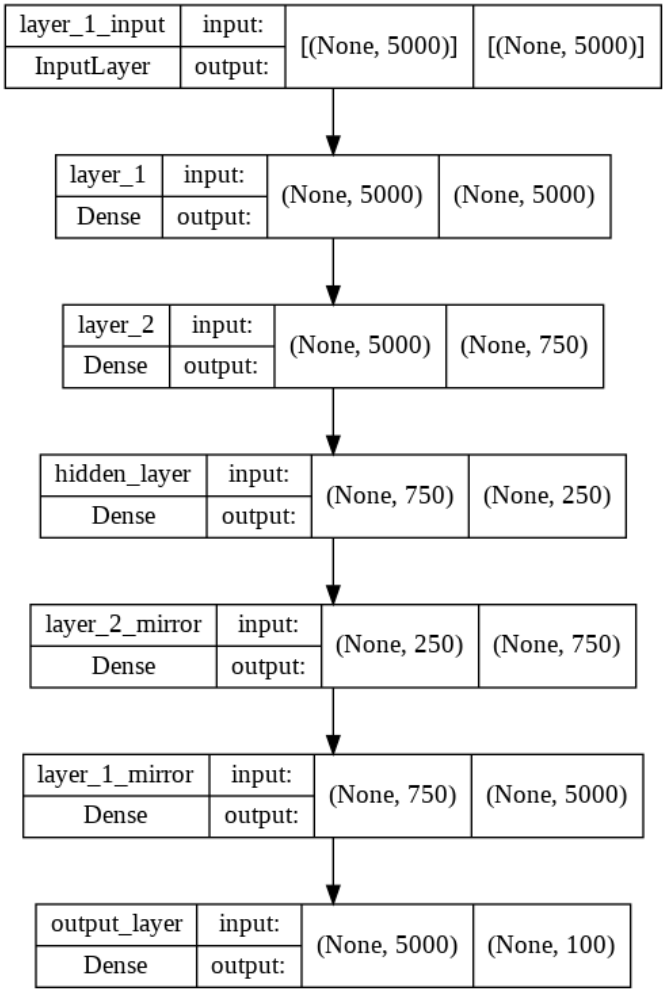

Fig. 4. the 2 layer SDAE model architecture.

We conducted our experiment by training our SDAE replication on the training data, and after each epoch testing our model on the testing data. We repeated this process for 100 epochs and saved the model on the epoch where it performed the best on the testing data. The testing accuracy for each epoch is shown below in figure 5.

This model was trained in a Google Colab Notebook environment. The system environment info is shown below in II

After training the SDAE model over 100 epochs, we saved the model when it reached its best accuracy on the testing dataset.

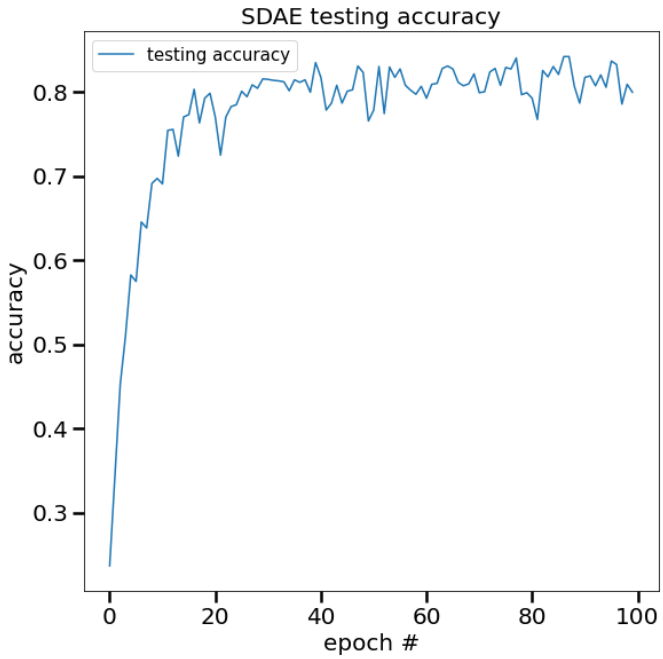

Fig. 5. Accuracy results over epochs trained.

TABLE II
ENVIRONMENT INFORMATION.

OS	Ubuntu 18.04.5 LTS
CPU	Intel(R) Xeon(R) CPU
RAM	15109 MiB
GPU	Tesla T4

D. Our method

In our method, we will also be using the SDAE architecture, but instead of just one single model, we will be using an ensemble of multiple SDAE models. An ensemble is a collection of many different models. Instead of just using one single model, we will use a collection of many different models where each model gets a vote on what the correct answer is [9]. Typically, ensemble models outperform singular models in almost every benchmark as seen through previous experimentation [9]. The ensemble method works because in many cases, a single model can't find a consistent relationship between the input and output. The different models will create different functions that show the relationship between the inputs and outputs. By using many different models, the ensemble algorithm can better predict the correct output because if the majority of the models agree on one single output using different functions, then there is a high probability that it is the correct output.

In our ensemble, we used many different SDAEs, each with a different 2nd layer output size and a different hidden layer output size. Each model is trained separately for 100 epochs, and the best epoch is saved and loaded so that each model will build their own relationship between input and output. Then, the models are brought together in an ensemble, where they each get a vote on what the correct output will be based on the input, and the prediction with most votes will be the ensemble's prediction.

The different models are shown below in Table III.

TABLE III
ENSEMBLE MODELS LAYER INFORMATION.

Model No.	layer 2	hidden layer
1	1000	1000
2	1000	750
3	1000	500
4	1000	250
5	1000	125
6	750	750
7	750	500
8	750	250
9	750	125
10	500	500
11	500	250
12	500	125
13	250	250
14	250	125

Our code is provided at https://github.com/Drime648/tor-deep-learning. We used the Tensorflow Keras python framework for all of our deep learning models.

V. RESULTS & ANALYSIS

Using this ensemble method, we were able to achieve the following results detailed in Table IV.

TABLE IV
ENSEMBLE RESULTS COMPARED WITH ORIGINAL SDAE MODEL.

Metric	SDAE Score	Ensemble Score	Difference
Precision	85.4%	88.9%	+3.5%
Recall	84.6%	89.3%	+4.6%
F1 Score	84.4%	89.2%	+4.8%

We have outperformed the SDAE in all benchmarks simply by increasing and varying the number of SDAE models. As [2]found, the SDAE model with a layer 2 of 750 and layer 1 of 250 had the best results, and all the other versions of the SDAE models under performed in the accuracy benchmark. Even though our models in the ensemble were individually worse than the single SDAE model, combined they were able to outperform the single SDAE model.

VI. DISCUSSION AND FUTURE WORKS

These results show that the deep learning website fingerprinting attack can be improved upon further through the use of ensemble training. This process can be replicated to any model, not just the SDAE. For example, the VarCNN [7] model can use this method to further increase its precision, recall, and F1 score. While experimenting with our model, we noticed that if we used too many models, our system crashed. We decided to be safe and only used 14 models, but we did not find the perfect number of models that we could use. We could also have used different number of neurons for the second layer and hidden layer, and leave that optimization for future works. One drawback of our model is that it is very slow. This is

because we use many models, so we have to train each model individually. While [2] only trained one model, we have to train 14 models. The SDAE model wasn't very complex, so this wasn't a big issue, but for big models like the VarCNN or if using bigger datasets, a stronger machine will have to be used when incorporating the ensemble method or else there will be extremely long training times.

A. Mitigations

The best probable mitigation for this attack may be the Walkie-Talkie [8] defense. The walkie-talkie defense alters the web browser from full-duplex communication with the user to half-duplex communication. This means that only one side can send messages at a time. This would render the website fingerprinting attack useless because each sequence will be modelled into a burst. The burst sequences will leak less information and the websites will appear to be the same in the eyes of a person and any machine learning algorithm.

VII. Conclusion

In this paper we proposed a method for improving a previous deep learning website fingerprinting attack using ensemble learning. The previous attack made use of an SDAE neural network that took inputs of cell directions in a communication with the Tor network in the form of 1, -1, or 0 for padding the vector. By increasing the number of models, we were able to increase the precision by 3.5%, recall by 4.6%, and the F1 score by 4.8%. This paper shows that previous deep learning attacks can be improved and be used in fingerprinting attacks to violate privacy in Tor communications.

It can be meaningful to use this method on other state of the art website fingerprinting attacks like the VarCNN model [7], which is a prospect we leave to future researchers.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation onion router," in *13th USENIX Security Symposium (USENIX Security 04)*. San Diego, CA: USENIX Association, Aug. 2004. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[2] 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.

[3] arma, "Improving tor's anonymity by changing guard parameters: Tor project," Oct 2013. [Online]. Available: https://blog.torproject.org/improving-tors-anonymity-changing-guard-parameters/

[4] N. McCullum, "Deep learning neural networks explained in plain english," Apr 2021. [Online]. Available: https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/

[5] 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.

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

[7] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *arXiv preprint arXiv:1802.10215*, 2018.

[8] T. Wang and I. Goldberg, "Walkie-Talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1375–1390. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[9] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, no. 1, pp. 110–125, 2002.