University of Bergen
Department of informatics

Deanonymizing communications on The Onion Router (TOR) network with Deep Learning

Author: Halvard Barstad
Supervisor: Chunlei Li

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

June, 2021

Abstract

Staying anonymous on the Internet is becoming increasingly difficult. Advertisements for services that to some extent offer privacy are frequent, and the general population seems to be more aware of privacy than previously. The National Security Agency documents leaked by Edward Snowden, revealed how the United States performed mass surveillance and described how it was close to impossible not to be digitally monitored by powerful governments. However, there was one specific type of technology that the NSA themselves internally acknowledged made surveillance hard. This type of technology is called onion routing, and the most popular software for this is The Onion Router, usually referred to as Tor. Onion routing works by sending packets through a chain of multiple nodes before it reaches its destination. Because of this, a node will only know the previous and the next node in the chain, making it impossible to know both who sent it, and where it is headed at the same time as long as the circuit consists of three or more nodes. For a good amount of years this has been considered the most secure way of staying anonymous, but with increasingly powerful computers, deep neural networks have become extremely effective at conducting multiple tasks. This thesis will explore how deep neural networks can be utilized to create digital fingerprints of websites, and then use these fingerprints to possibly identify which page a user is trying to access. The source code of the project is available at https://github.com/HallyB96/TOR-Deep-Fingerprinting-Master-Thesis.

Preface

This thesis is a conclusion of my two year master's degree in Secure and Reliable Communications at the University of Bergen (UiB). Prior to my master´s degree, I finished a bachelor's degree in Computer Security, also at UiB. Much of the knowledge used to produce this thesis has been learned through the courses I have taken, and some has been learned out of my own interest and while working on the thesis. The process of writing this thesis started on the 15th August 2020, and ended on 1. June 2021. I would like to thank my supervisor Associate Professor Chunlei Li at UiB for his excellent help in supervising during this period.

Halvard Barstad
01 June, 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

On the internet, privacy is an important topic. Digital footprints are left everywhere and gathered by data collection through various different techniques. One increasingly popular service is Virtual Private Networks (VPNs), as they add another layer of encryption on the communications. VPNs do however not prevent someone from seeing who you are, and what pages you are visiting. The Onion Router (TOR) provides a service where the data has multiple layers of encryption, and removes the possibility for someone to register what pages a specific user is visiting. Tor is because of this the gateway to what is typically called the dark web, or the deep web. This is a part of the Internet where users and servers can stay completely anonymous if used correctly. A question often asked is why would someone need to hide themselves this thoroughly? It is true that a huge downside of this technology is how it enables criminals to communicate without fearing criminal prosecution, but there are also other people requiring this level of anonymity, such as some journalists and people living in oppressed societies.

For a while Tor has been considered very safe to use as even the National Security Agency (NSA) unintentionally leaked that they have been unsuccessful in mass deanonymization of users [22]. However, more recently machine learning, and with that deep learning, has had huge improvements in performance opening a world of new attacks, including the fingerprinting attack to be explored in this thesis [47].

Fingerprinting is traditionally a term used to describe various techniques used in order to create a digital fingerprint of a website, server or a device. The fingerprint aims to be unique to that service, like fingerprints to human beings. The procedure is initially performed by tracing large amounts of packets from the source. Information such as time sent, time received, packet length, and more is then stored and used in order to attempt to create a digital fingerprint unique to the source. Using the data to create this fingerprint can be done in multiple ways, but recently the methods having most success are using deep neural networks which are great at handling large amounts of data, and finding patterns that can be used to uniquely define these fingerprints. In the case of Tor, having fingerprints of a set of websites allows someone to eavesdrop on a user's incoming and outgoing communications, and by that possibly identify what server or web page the user is communicating with [47].

On the Internet, fingerprinting is more traditionally done on devices. By creating fingerprints of end-user's devices, such as an individual's browser, one could be able to track a user's online activity. The concept of creating the fingerprint remains the same, but slightly different data would be used such as resolution size, browser type, and more. This opens the possibility of potentially identifying someone without having that entity ever personally identifying themselves [23].

On the standard Internet, digital fingerprinting does open for quite intrusive possibilities, but it has possibly even bigger consequences for Tor users, where anonymity is the key concept.

1.2 Problem Definition

The focus in this thesis will be how new and modern deep neural networks may be used to deanonymize Tor users. This may not be a full implementation of a program which performs these tasks, but rather research into at which extent such an attack would be possible. In order to do this research, some implementations will be necessary. The main topic of the thesis is defined as follows:

How can deep neural networks be used to create digital fingerprints in order to deanonymize TOR users?

The thesis will produce a detailed description of the work done through implementation and different programs used to perform the various tasks. There are several existing datasets containing Tor traffic which can be used to train the neural networks. The plan is to use these datasets and see what results can be achieved and reproduced from previous work. The obtained results will then be deeply analyzed and discussed.

Following the implementation of the attack in a basic closed world setting, further steps have to be made to create a scenario as realistic as possible. It is at this stage necessary to evaluate in an open world setting to further strengthen the realism behind the results, and also by comparing these results to those previously obtained in a closed world setting. This will be achieved through using datasets of different structures, and more detailed evaluation methods. Additionally, two different proposed defense mechanisms will be tested in order to see if they provide the extra level of security that they intend.

When proper evaluation of the attack and its proposed defense mechanisms is completed, the final phase is to look further into what possibilities these types of attacks open. Maybe there could be room for further improvements, and also what implications does this attack actually have?

The objectives to be solved are:

- Implement classifiers on existing data from previous work to achieve same or better results.
- Run the attack on data defended by different defense mechanisms.
- Evaluate the classifiers in different scenarios to achieve more realistic results.
- Consider more theoretical approaches and what implication the attack has in the real world.

1.3 Intended Results

The work in this thesis aims to result in more insight into how modern techniques can be utilized to deanonymize Tor users. As Tor is considered safe, it is not necessarily realistic to believe that major security issues will be discovered. It is instead more likely that the work will serve as further confirmation that Tor still lives up to the security and privacy expectations it has. With that in mind the goal of the thesis is to achieve results closely representing the real world, and provide insight into what this could mean for the security of Tor.

As simulating an attack like this is hard due to technological restrictions, a larger entity such as a governmental agency could be a lot closer to actually performing such an attack. This makes it of interest to discuss what theoretical attacks could be possible under some circumstances. By imagining theoretical settings and exploring these, there might be questions arising around what different policies would mean if they should ever be implemented.

1.4 Related Work

Fingerprinting attacks on Tor have through the years had some contributions, with the more recent ones being of most relevance.

These types of attacks have long been theorized about, and the first research to be made on the topic was in 2009 by Hermann et al [24]. They did not achieve particularly good results with only 3% accuracy. Fingerprinting attacks were then by some not considered to be a threat to Tor.

Over the years further research was conducted, with quite a large increase in attack accuracy. The largest breakthrough came in late 2018, when the research paper "Deep Fingerprinting: Undermining Website Fingerprinting Defensed with Deep Learning" [47], showed how modern neural networks can be used to achieve high accuracy when predicting which sites users were visiting, even when the communications had modern defense mechanisms in place. This paper produced by Sirinam et al. sparked new light in the debate around fingerprinting attacks, and how feasible they could be.

1.5 Thesis Outline

The structure of the thesis is organized in the following chapters:

Chapter 2: Theoretical Background
In chapter 2 the thesis aims to give the required theoretical knowledge necessary in order to understand how the attack is performed. The topics explored include cryptography, the Tor design and its history, fingerprinting attacks and defenses, and lastly some theoretical machine learning with neural networks being of the primary focus. While some of the smaller topics might not be directly mentioned in the following chapters, they are of relevance for the complete understanding.

Chapter 3: Experimental Study
In chapter 3 the implementation and evaluation of the attack is described in detail. The chapter initially presents the different scenarios to be considered, as well as justification to why different decisions were made. In the evaluation section, the models are analyzed and compared, including the different scenarios with different defense mechanisms. Technical justification of the results are also made. The discussion section aims to present a reasonable justification around what the attack is realistically able to do, as well as what implications this attack might have in the real world.

Chapter 4: Conclusion and Discussion
Chapter 4 will present a final conclusion based of the previous chapter, and also present different possibilities of future work.

Chapter 2

Theoretical Background

2.1 Cryptography

The purpose of cryptography is to keep communication between two or more parties secure and hidden to third parties, which the content is not intended for. Traditionally cryptographic ciphers were symmetric, meaning the key used for encryption and decryption was the same. Symmetric cryptographic ciphers have a long history, and the ciphers such as the famous Caesar cipher was invented by the ancient Romans and involved shifting characters in the alphabet [39, p. 18]. Today the symmetric ciphers have become more advanced, and usually consist of a series of bit-operations, like in the Advanced Encryption Standard (AES). The benefit of this is that it can be implemented directly in the hardware which makes it fast.

While symmetric encryption in modern times is still widely used due to its computational speed, it requires some additional protocols. In order for symmetric encryption to work, the parties involved need to agree on some shared predetermined secret, also referred to as a shared secret key. In contrast to symmetric encryption, asymmetric methods instead use different keys during encryption and decryption, often referred to as a key pair, which opens some different possibilities. As it is infeasible for the parties to physically meet, and agree on a key, we instead use asymmetric protocols that let the parties generate shared secret keys without requiring previous knowledge of each other. Several asymmetric encryption

protocols originated during the 1970's proposing solutions to this challenge, and the name public key cryptography became widely used. The protocols used involved having a public key and a private key, where the public key would be open to anyone, and the private key would be kept secret to the owner. Two of the most widely used asymmetric protocols include the RSA cryptosystem, which allows parties to communicate encrypted with both having a public private key pair, and the Diffie-Hellman protocol (DH) which allows two parties to generate a shared secret to use for further symmetrically encrypted communications, digital signatures schemes (DSA), and their elliptic curve variants [39, chp. 1, 6].

2.1.1 RSA

RSA (Rivest-Shamir-Adleman) is a public key cryptography protocol and perhaps the most famous to date. It was developed in 1977 and solves a lot of the same challenges as other asymmetric protocols, and does so by using a hard mathematical problem known as "The Integer Factorization Problem". Given two large primes, p, q, multiplying these two primes to find the number $n = p * q$ is easy. If however only n is publicly known, factoring n into the two primes p, q is very hard if the primes are large enough [44]. The key generation works as follows:

Output: public key: $K_{pub} = (n, e)$ *and private key* $K_{pr} = d$

1. Choose two larges primes p, q
2. Compute $n = p * q$
3. Compute $\phi(n) = (p-1)(q-1)$
4. Select public exponent e such that $e \in \{1, 2, ..., \phi(n) - 1\}$ such that $\gcd(e, \phi(n)) = 1$
5. Compute private key d such that $d * e \equiv 1 \mod \phi(n)$

Given the plaintext x, the ciphertext c can be obtained by the calculation $c \equiv x^e \mod n$, and given the ciphertext, the plaintext can be obtained by the calculation $x \equiv c^d \mod n$ [39, chp. 7].

2.1.2 Diffie-Hellman Protocol (DH)

Diffie-Hellman key exchange (DHKE) was in 1976 proposed by Whitfield Diffie and Martin Hellman as the first practical asymmetric protocol, and it provided a solution to the challenge of distributing keys [12]. The underlying mathematical problem used in DHKE is shown to be equivalent to the following problem:

Discrete Logarithm Problem (DLP) in $\mathbb{Z}_p^*$

Given is the finite cyclic group in $\mathbb{Z}_p^*$ or order $p-1$ and a primitive element $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$. The DLP is the problem of determining the integer $1 \leq x \leq p-1$ such that $\alpha^x \equiv \beta \mod p$

Based on the discrete logarithm problem, the DHKE protocol first requires a set-up phase where a large prime p is picked, and an integer α. Both values are then published, and are often referred to as the domain parameters. We can then consider a key exchange between the two parties Alice and Bob.

Alice starts by choosing a value a which is kept private. She then computes $A = \alpha^a \mod p$, and sends it to Bob. Bob performs the same procedure as Alice by choosing a value b, which is kept private to Bob. He then computes $B = \alpha^b \mod p$, and sends it to Alice. Both Alice and Bob can now compute a shared secret. Alice takes B received by Bob, and computes $K_{AB} = B^a \mod p$, and Bob computes similarly $K_{AB} = A^b \mod p$. The reason this works is because

$$B^a \equiv (\alpha^b)^a \equiv (\alpha^a)^b \mod p$$
$$A^b \equiv (\alpha^a)^b \equiv (\alpha^a)^b \mod p$$

and by this method Alice and Bob now shares a secret key K_{AB} without requiring Alice or Bob to have any previous knowledge of each other. This secret key allows Alice and Bob to encrypt their communications with some symmetric protocol [39, chp. 8].

2.1.3 DLP-based encryption and signatures

Continuing with the example from previously in chapter 2.1.2, Alice and Bob can also use the shared secret key K_{AB} for further asymmetric encrypted communications with the Elgamal cryptosystem, or to provide signatures with the Digital Signature Algorithm (DSA) or elliptic curve DSA (ECDSA).

In order for Alice to send a message x to Bob, she can encrypt through using the Elgamal cryptosystem, by computing the ciphertext c

$$c \equiv x * K_{AB} \mod p,$$

and send c to Bob. Bob can then recover the message x from c by computing

$$x \equiv c * K_{AB}^{-1} \mod p$$

Bob will then have recovered the message, and can read its contents in plaintext. However, this only shows the basic principle of the Elgamal scheme, and in order for this to work, Alice has to generate a new public-private key pair for each message. For the actual encryption, Alice multiplies the plaintext x by a masking key K_m. For decryption, Bob can then can decrypt by multiplying with the inverse mask, and is able to do so since the masking key is created with Bobs own public parameters [39, chp. 8.5].

In a different scenario, Alice might instead want to sign a message in order for others to verify her identity. This can be done with the DSA algorithm [39, chp. 10,4], where two cyclic groups are used. Alice can then create a signature by using both her public key and private key. When Bob wants to verify the signature, he can verify it by only using the public key provided by Alice. This means that Alice proves to Bob that she has the private key corresponding to the public key [39, chp. 10,4].

Another option is to use ECDSA, which instead uses the discrete logarithm problem constructed in the group of an elliptic curve, making the actual calculation quite different to DSA. The main advantages with elliptic curves is how they have shown to be secure against certain attacks, making the required bit-size lower than in the case of other asymmetric cryptosystems [39, chp. 10,5].

2.1.4 Public Key Infrastructure (PKI)

While public key cryptography does not require a secure channel, the authentication of user's public keys are of paramount importance. When Alice receives a public key from Bob, she has no way of knowing if the key is in fact from Bob, or someone pretending to be Bob. This is known as a man-in-the-middle attack, where an adversary is pretending to be the communicating parties, and by that compromising the security of the communications. Without proper authentication, an attacker can pretend to be Alice when communicating to Bob, and pretend to be Bob when communicating to Alice, without Alice or Bob ever knowing they were tricked.

In order to solve this problem, public key infrastructure is introduced, and it works by adding layers of trust to the system through certificates. A certificate can simply be considered as a public key, and a signature with the private key. If anyone should try to change the public key on the certificate, the signature would no longer be valid, and vice versa. Certificate authorities (CAs) are entities that provide extra layers of trust to the fact that someone's public key is the correct one. CAs can typically be large companies, Internet service providers (ISPs), or even governments. Together they provide authentication through a chain of trust where a certificate is signed by one or more layers of CAs [39, p. 342]. While this approach is considered the best option available for the Internet, it is still not completely fail-safe, and there have been situations where CAs provided illegitimate certificates [11].

2.1.5 Transport Layer Security (TLS)

The TLS protocol is today the most popular protocol that provides secure communications. TLS combines symmetric and asymmetric encryption together with PKI and more, which makes it a protocol that contains all the necessary steps to set up a secure channel between a client and a server. The setup phase is called a handshake, and is performed through several steps.

A simplistic workflow of the TLS protocol is a follows. First the client starts by sending a "ClientHello" to the server. The ClientHello contains a list of different cryptographic ciphers supported by the client, as well as a "client random" which is a random string of bytes. The server then responds to the client with a "ServerHello", which contains the

server's certificate, a cipher chosen from the list of ciphers provided by the client, and a "server random", which is also a random string of bytes.The client then verifies the server's certificate with the certificate authority, and confirms that the server is who he claims to be. The client then sends another string of random bytes known as the "premaster secret". This string is encrypted with the server's public key, meaning only the server can decrypt it with its corresponding private key. The server then decrypts the premaster secret, and both the server and the client generates shared secrets by using the server random, client random and premaster secret. Both then send a "finished" message to each other to signal that the handshake is complete, and communication may continue using the shared secret keys with some symmetric encryption algorithm [3].

2.2 The Onion Router - Tor

The mentions of Tor dates back to 1995 when the US Office of Naval Research along with Defense Advanced Research Projects Agency (DARPA) worked on developing a new type of technology which would make it hard to trace traffic back to them. The idea was based on passing the traffic through random nodes before reaching its destination. The point of this was to create confusion around who the sender and the intended destination was, and by that making it hard to identify the end-entities of communication. The purpose for developing this technology was originally not to strengthen privacy, but to create possibilities for undercover personnel to communicate anonymously without fearing capture.

However, after staying in development for several years the realization came that passing traffic through military controlled nodes only, would not be good enough, and a more mass-supported network consisting of several thousands of volunteers was instead necessary. Because of this, Tor was in late 2004 released under an open source license, and the technology was now available to the public [33]. Tor is now primarily maintained by The Tor Project which is an non-profit organization of computer scientists.

2.2.1 Onion Routing

Onion routing is an attempt at making communications anonymous by adding multiple layers to the packets in the same way an onion has multiple layers. Normally on the Internet, communications are made between two entities by providing destination and source ip-addresses

to packets in similar fashion to a mailing service. Most data is also encrypted, which makes it hard for some eavesdroppers to determine exactly what the two entities are transmitting to each other. However, an eavesdropper can easily know that the entities are communicating, which could already be considered a security issue by some. Onion routing tackles this issue by instead introducing some extra nodes in the communication circuit which the packets pass between before reaching the destination. When a user wishes to visit a specific website, he will first set up a circuit. This circuit consists of multiple relay cells, which the user generates a shared secret key with. Each relay cell has different symmetric keys shared with the user. When the user then wants to send data through the circuit to the end goal, he starts off by encrypting the data with the key from the last cell, and iteratively encrypts the packet with the key from the second to last relay cell, and so on until the packet has a layer of encryption for each relay cell in the chain [17].

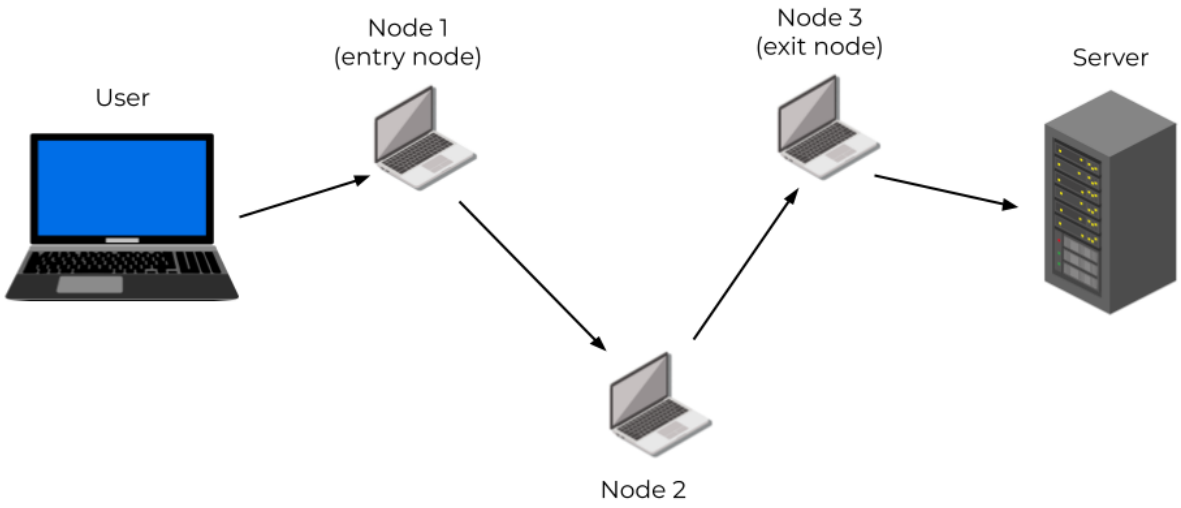


Figure 2.1: Figure displaying the nodes between the client and a server.

Consider a circuit with n relay cells, with a packet p encrypted multiple times with symmetric keys k_i shared between the user and the relay i, resulting in a final ciphertext C which is then forwarded through the circuit:

$$C = E(K, p) = E_{k_1}(...E_{k_{n-2}}(E_{k_{n-1}}(E_{k_n}(p))))$$

When the packet then is forwarded through the relay cells, each cell then decrypts the packet with its symmetric key shared with the user, before it forwards the packet to the next IP-address in the chain. By doing this, onion routing creates a setting where anyone who eavesdrops on the communications at any point in the process of a packet being sent, will not be able to both determine who both end-points are, at the same time as each node in the chain will only know its previous and next node. This does not mean that it is completely anonymous. Someone eavesdropping at the first node will know who the user is without knowing what the user is browsing. Furthermore, eavesdropping at the exit node will reveal what destination is being requested as the packets travelling from the exit node to the server will be decrypted. This means that without careful usage, a user might reveal their identity if the communication between the user and the server are not end-to-end encrypted. There also exist correlation attacks on Tor where for example some authorities might be suspicious of a person's Internet activities. By then monitoring when that person is active on his computer, and looking at traffic flowing through the exit nodes, the user could potentially be identified and completely de-anonymized [52].

2.2.2 Cells

Packets sent on the Tor network are typically referred to as cells. Each Tor cell has a fixed length of 512 bytes, and consists of both a header and a payload. The header is necessary as it consists of a circuit ID which is used to identify which circuit a given cell is belonging to. This is important as multiple circuits can be multiplexed over one TLS connection. Also in the header is a command field which describes what to do with the payload of the cell. The cells can be categorized into either control cells, or relay cells. Control cells are used to forward the commands padding, create or destroy, where padding can be used for keepalive and link padding, create is used to set up a new circuit and destroy is used to tear down a circuit. Relay cells are the cells containing the actual data headed for the end-points of the stream. These cells have an additional header called the relay header which contains a stream ID in order to multiplex multiple streams, a checksum for end-to-end integrity check, a payload length field and a relay command. Similarly to the control commands, these relay commands serve many of the same purposes, but also have a few more commands such as extend in order to extend a circuit by one relay, as well as commands for both opening and closing a circuit and sending notifications over the circuit [13].

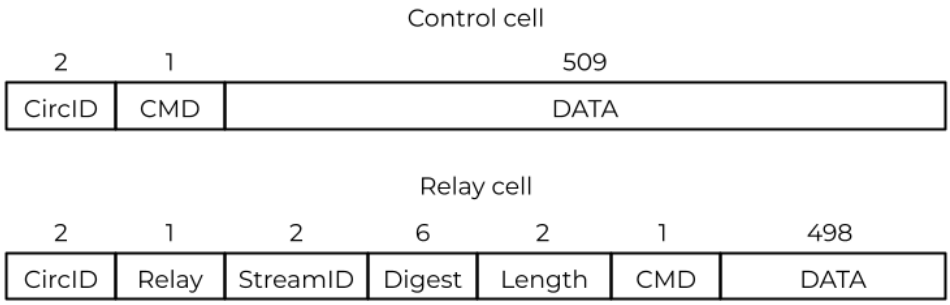


Figure 2.2: Figure displaying structure of different cells in Tor.

2.2.3 Constructing and closing a circuit

Constructing a new circuit is a process performed incrementally with all the nodes of the circuit. The user first sends a *create cell* command to the first node of the path. This cell contains the first required part for Diffie-Hellman key exchange protocol g^x, where g is a generator, typically a finite cyclic group of order n agreed upon by both parties, and x is a random number picked by the user where $1 < x < n$ .The first node then responds to the user with g^y and the hash of $(g^y)^x$ where g is the same generator, y is a random number picked by the node, and $(g^y)^x$ is computed by combining what the user sends to the node g^x along with the nodes own y. The user can now also compute $(g^x)^y$ and compare its hash to the hash sent by the node, and see if it matches as $(g^x)^y = (g^y)^x$ [39, chp. 8]. With this step completed, the user has negotiated a shared secret key with the first node in the circuit, and now has to repeat the procedure with the remaining nodes. However, it is not performed directly with the next nodes, as this would compromise the anonymity of the user by exposing their identity to the next nodes. Instead, the first node is used as a relay to the second node, and assists in the process of generating a shared secret with the next node. This is done by the user sending a *relay extend* command to the first node and specifying the address of the next node. When the user has generated a shared secret with the second node, it is sent from the second node to the first node with a *relay created* cell, and the first node sends it to the user with a *relay extended* cell. This is then repeated with the third node with the same procedure until the user has a shared secret symmetric key with all the nodes in the circuit to use for further communication secured by TLS.

To close a circuit, the user can send the control cell *destroy*. Upon receiving a *destroy* cell, the node will close all streams on the corresponding circuit. This process can also be

performed incrementally with all the nodes similarly to the when constructing the circuit. This is done by the user sending a *relay truncate* cell to a node in the circuit. This node will then forward a *destroy* cell to the next node before it is acknowledged back to the user with a *relay truncated* cell [13].

2.2.4 Hidden Services

Tor hidden services, often referred to as responder anonymity, addresses a series of challenges that a server might face. Hidden services primarily allow for the ip-addresses of servers to not be public, as this would expose the server's identity. In the design, multiple goals were selected. A server should be robust against denial of service attacks. The server's identity must also remain hidden in case of router failure which in practice requires the server to be able to connect to multiple routers. Attackers should also not be able to pose as a server in order to disrepute the actual server, or perform illegal activities while posing as the server [13].

Server anonymity is in Tor achieved by adding introduction points and rendezvous points. Introduction points are picked by the server, where the server signs his advertisement with his public key. A circuit is then built between the server and the introduction points. The client chooses an onion router as its rendezvous point and gives it a random cookie, before connecting to one of the introduction points selected by the server. The client would not know where to connect to the server unless the server itself at some point has revealed that it exists, and where to connect to it. The client then informs the server of its rendezvous point, starts a DH handshake and passes the random cookie, and then awaits for the server to connect to the clients rendezvous point. This gives the server the option to decide whether or not it wants to open a connection to this client. If the server decides to connect to the client, it opens a circuit to the rendezvous point, completes the handshake verified by the random cookie and communication may then begin in a setting where both the client and the server are completely anonymous [13].

2.2.5 Weaknesses of Tor

While Tor has gone to great lengths in ensuring anonymity and security for its clients, it still has some flaws that could potentially lead to certain types of attacks. Lack of end-to-end encryption could be considered one of these, but is not unique to attacks on Tor, and because of this not considered a weakness of Tor itself.

Correlation attacks have been a long threat to the anonymity of Tor, and have been a methodology with some successful attacks over the years. In 2013, a Harvard student was arrested for making bomb threats. The officials were able to determine that a computer on the university network had been used to connect to Tor, and by that were able link the suspect to the threats made [8]. While this is a type of correlation attack that is rather easy to avoid, more sophisticated correlation attacks have gained attention throughout the existence of the Tor network. Research has been done in order to find out how many relays an attacker needs to control in order to successfully notice when traffic that correlates to each other is passing through different relays. If an attacker can control a set of relays, where some are entry nodes and some are exit nodes, there might be a chance that users construct circuits that pass through nodes controlled by the attacker. Furthermore, someone having control of IXPs (Internet exchange points), may be even more successful at identifying correlated traffic [28].

Throughout the years, software vulnerabilities have also existed. In 2013, Firefox 17 had a security vulnerability that enabled the FBI to run malicious JavaScript as a Windows executable file, which resulted in the possibility of the client leaking both its IP address and MAC address [5]. This led to a server sharing illegal content being taken down, and its owners arrested. More recently, fingerprinting attacks have gained increased interest as the effectiveness of machine learning and especially neural networks have greatly increased. Utilizing these new techniques, new attacks are coming to light, and might be the type of attacks that require attention and research [47].

2.3 Digital Fingerprinting

Fingerprinting attacks is a genre of attacks where an attacker attempts to construct a digital fingerprint unique for a specific user or a website. This means that fingerprinting attacks can be separated into two categories: Browser fingerprinting, and fingerprinting targeting servers.

Browser fingerprinting is performed against a single user's browser with the intention to recognize a user's identity without requiring personal information of the user. This can be achieved by finding unique patterns in the user's browser that could be used to produce a unique fingerprint. These patterns can emerge as browsers have multiple settings, which when considered together have a very low chance of two or more having the exact same settings. The settings considered are usually resolution, time-zone, operating system, plugins, and more [23, 50]. Research done by the Electronic Frontier Foundation showed that only 1 in 286 777 browsers share the same fingerprint [14].

In the same way as browsers can have similar patterns, digital fingerprints of individual websites can be made as they also have characteristics that when combined forges unique fingerprints. An example of this is how different webpages differ in amount of data, which results in the total amount of bytes sent over one or multiple packets to be different. Loading the same website at different times would lead to similar looking packet traces if the contents of the website is not changed too much [47, 41, 24].

Traditionally on the Internet, data about the users can be stored in cookies, but they can be deleted, and also requires the user to agree to them being used in the first place. Browser fingerprinting then becomes an alternative for corporations to track and learn about the patterns of users without the users ever knowing or agreeing to it. While this is the main concern on the regular Internet where IP-addresses are publicly shown, the consequences for the Tor network are much larger as website fingerprinting could allow an attacker to know what content a specific user is browsing without monitoring the entry-node and exit-node at the same time. To defend against such attacks on Tor, some defensive mechanisms have been proposed which aims to make it hard to identify the individual pages by obfuscating patterns in packets through different methods.

2.3.1 Walkie-Talkie

Walkie-Talkie is a defensive measure against website fingerprinting which consists of two important components: half-duplex communication, and burst molding. By combining the two, the packets in a sequence become exactly the same with regards to time, length, direction and ordering.

Half-duplex communication is a mode where browsers are only able to either receive or send at some point in time, in contrast to full-duplex which can both send and receive at the same time. Both incoming and outgoing packets are sectioned into burst sequences, sent in bursts, and later molded at a low computational cost. By employing half-duplex mode, it removes patterns in the sequences that could occur by having certain features on a web page load at specific times and by that, creating a more distinct pattern in the packet sequence.

Burst molding is inspired by the concept of having a decoy in order to make it impossible for the attacker to know which site is actually being visited. However the problem with this defense mechanism is that it approximately doubles the number of bytes sent that are not application content, also called dummy cells. Walkie-Talkie instead simulates loading the two pages by loading the supersequence of two burst sequences. As defined by the creators, a sequence $s^{'}$ is a supersequence of s if $s^{'}$ contains s [55]. Burst molding then adds fake cells to i burst sequences b_i where each sequence $b_i = (max\{b_{i+}, b'_{i+}\}, max\{b_{i-}, b'_{i-}\})$ where b'_i denotes burst sequence i of the decoy page, $+$ denotes outgoing and $-$ denotes in-going packet direction. In the case where the number of burst sequences are different, fake random packets are added to the shorter sequence. This then leads to the scenario where rather than completely loading two pages, burst molding instead uses the max. An attacker will only know that there exists a sub-sequence of the loaded super-sequence that corresponds to the real page, but has no way of actually determining what this sequence is. Information about potential decoy pages are necessary to store, but does not require storing the ordering of cells which older methods required. In the paper published by the authors of Walkie-Talkie, this difference was shown to be 20 bytes of extra information, instead of 36kB for each decoy page. Decoy pages can also be picked

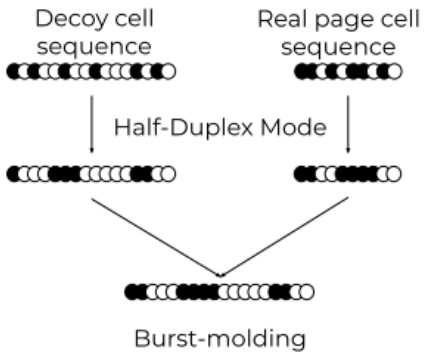


Figure 2.3: Burst molding decoy and real page

cleverly in order to avoid unnecessary computation by using the same pair of sequences each time when visiting one of the two pages, and not having to burst-mold the sequence every time. If we then consider the security of this defense, we have a client choosing a site with probability p, and a decoy page with the probability $p^{'}$, making the attackers precision of guessing the correct site never exceed $\frac{p}{p+(1-p)p'}$. Also the theoretical maximum precision is 50% as the attacker at some point will have to guess between the two pages used in the sequence [55].

2.3.2 WTF-PAD

WTF-PAD is another technique developed to prevent website fingerprinting attacks. The name WTF-PAD is short for *"Website Traffic Fingerprinting Protection with Adaptive Defense"*, and its aim is similar to Walkie-Talkie with regards to making the sequence of data become less recognizable. It tries to achieve this by using two different state machines, one for sending data, and one for receiving. These state machines have two states: burst mode and gap mode, but for simplicity we will only consider the receiving state machine going forward.

Burst mode is initiated when a real packet is received. It then starts a countdown until a new real packet is received, or the timer runs out. If a packet is received, the process starts again, or if the timer runs out, a dummy message is sent and the state machine switches state to gap mode. The time which determines how long to wait before a potential dummy message is sent, is governed by a histogram built based on a large dataset of web traffic, out of which delays between bursts are looked at.

In gap mode, the state machine uses another histogram to determine times at which dummy messages are sent. This histogram is also built based on a large dataset of web traffic. This makes gap mode send dummy messages at intervals which is typical of web traffic. Switching mode back to burst mode occurs when a real packet is received, or by some probability.

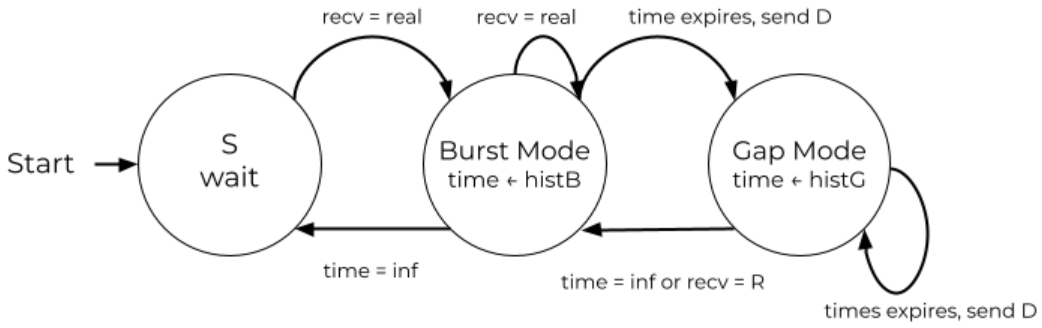


Figure 2.4: WTF-PAD receiver state machine

WTF-PAD also introduces control messages which can be forwarded from the client to the server in order for the client to have full control over the distribution of histograms used by the server. Also, the server can alert the user if relays in a circuit are sending with a padding that could mean the circuit is being exploited. Control messages are also used in order to start the transmission in a controlled fashion. Transmission is then stopped with a probability similar to the probability of switching from gap mode to burst mode without receiving a real packet [30].

2.4 Machine Learning - Deep Neural Networks

Machine learning is a relatively new and modern type of artificial intelligence that brings forth some alternative ways of solving specific types of problems [36, chp. 1]. While classically a program takes an input and produces the same output every time, machine learning algorithms try to improve automatically over time through experience. This is done by building and training models on a type of data with the goal of making the model generalize well to new and previously unseen data [16].

Machine learning can be divided into three different categories. Supervised learning is when models are trained on labelled data. The most typical problem in this category is image recognition and object detection. Unsupervised learning is when the data have no labels, and the model instead usually attempts to group similar objects into clusters, and by that is able to identify objects that do not clearly fit into one of the clusters [35]. Lastly we have reinforcement learning in which the model is working in an environment where it is rewarded for actions that are good, and its overall goal is to maximize its reward. These types of models are typically used for technology such as self driving cars where data is not labelled, and you instead wish to make the car act as a perfect human driver by rewarding it for good behaviour [31, 21].

With the overall goal being to generalize well to unseen data, a few different scenarios may occur. One case is when both the training and test accuracy of the model is low. This is usually a result of the model design not being complex enough. A more common case is when the model has great training accuracy, but fails when being fed unseen data. This scenario is called overfitting, and happens when the model is too closely fit to the training data, resulting in the model failing to generalize well. When developing machine learning models, this is something that must be taken into consideration, and that requires careful action to reduce the risk of occurring [39, chp. 2].

2.4.1 Neural Networks

One of the most popular types of models in modern machine learning is neural networks, as they have grown to become extremely powerful with the production of newer and more powerful hardware. Neural networks are inspired by the topology of the human brain with its many neurons.

A neural network consists of an input layer, hidden layers, and an output layer, with each layer consisting of multiple neurons. Each neuron has a weight which is frequently updated based on features the network considers important, and through that learns over time by repeatedly tweaking the weights of the many neurons.

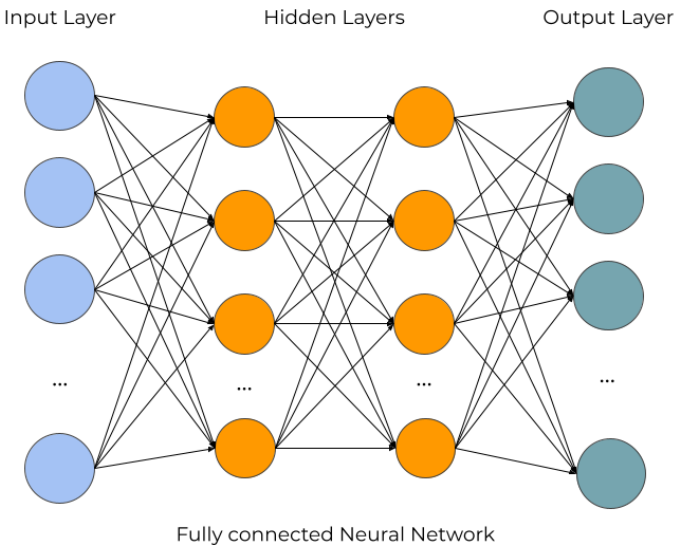


Figure 2.5: Basic illustration of a Neural Network

When training a neural network, data is being fed to the input layer. This data could for example be the pixel values for pictures of different species of animals, for a model that aims at recognizing animal species. The network then both forward and back propagates in order to learn.

2.4.2 Forward Propagation

Forward propagation is used to feed data through the network from the input layer to the output layer. This is done by computing the dot product of the incoming data values and the assigned weight before applying an activation function to it. Let ϕ represent the activation

function, b a bias term, $X_1...X_m$ the the input values, and $w_1...w_m$ the assigned weights of the inputs. With this we can determine the output signal from a neuron by the formula

$$\phi(\sum_{i=1}^{m}(w_i * x_i + b))$$

This output then acts as the input to the next layer where the process is repeated until we get the final output vector y from the output layer [6]. To check the performance of the model, we check how close to the the predicted value y was to the actual value through a loss function. A common loss function used is mean squared error defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

where n is the sample count, y is the actual value and $\widehat{y}$ is the predicted value [34]. The activation functions used can be different, and it defines the output from a node given an input. A common activation function is Rectified Linear Units (ReLU) defined as [53]:

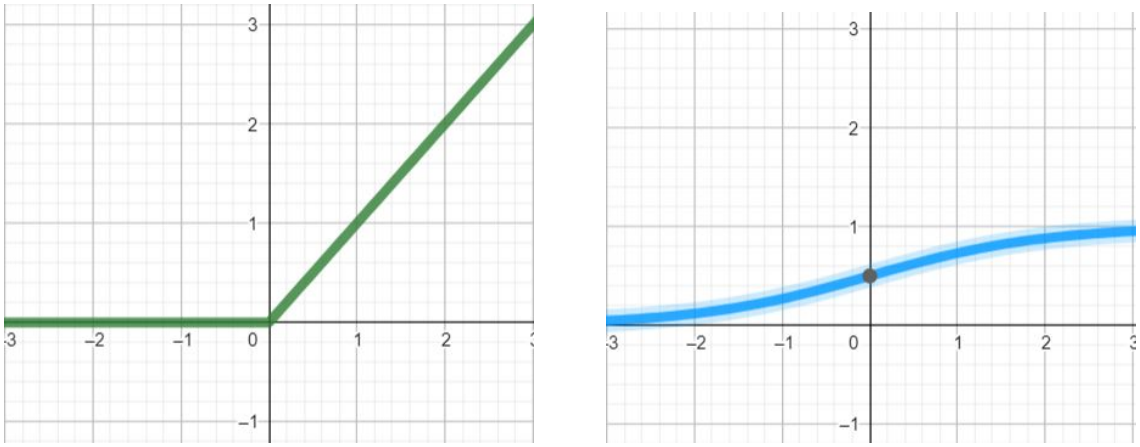
$$Z = b + \sum_i x_i w_i \qquad y = \begin{cases} Z, & \text{if } Z > 0 \\ 0, & \text{otherwise.} \end{cases}$$

As we see from the formula and plot 2.6a , ReLU always returns a value equal to or larger than zero. If it is necessary to capture negative input values, other activation functions can be used such as the Sigmoid function defined as [53]:

$$Z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-Z}}$$

As we can see from figure 2.6b, Sigmoid returns different values when given values smaller than 0, in contrast to ReLU.

After calculating the loss function from the output layer at the end of forward propagation, the information is fed back through the network through backpropagation.



(a) ReLU Function (b) Sigmoid Function

Figure 2.6: Plots displaying both activation functions

2.4.3 Backpropagation

The overall goal is to minimize the loss value of the network, by adjusting the weight and bias values repeatedly through backpropagation. The amount of adjustment done to the network's weights and biases is determined by the gradients of the cost function. Computing gradients are necessary as they tell us which direction a function is going, and by that can tell us how much a given parameter has to change in either direction in order to minimize the loss function, which is inversely proportional to the cost function C. The purpose of gradient descent is to work towards the global minima [6].

$$C = \frac{1}{2}(\widehat{y} - y)^2$$

When computing gradients, the networks weights vector w and biases vector b are initially randomized. A learning rate e is also introduced and it determines how much influence the gradient should have on the changes. The algorithm then runs until the termination condition is met.

$$w = w - e\frac{\partial C}{\partial w} \qquad b = b - e\frac{\partial C}{\partial b}$$

Derivatives of C calculated using partial derivatives.

With this being the general idea behind backpropagation and the training of a neural network, in practice it is performed with some additional ideas [32].

Batch gradient descent is a method where all training data is considered at each step. This is done by taking the average of the gradients of the training steps, and using the mean of this value to update the parameters of the network. This works great for models built on relatively small sets of data, but can become more problematic as the dataset size increases.

Stochastic gradient descent is another method that instead takes a single example which it feeds back to the network. This gradient is then computed and used to update the parameters of the network.

A final common method is mini batch gradient descent. This method combines both batch gradient descent, and stochastic gradient descent by selecting a fixed sized batch of training examples. The size of the batch is then smaller than the dataset, therefore mini-batch. The same procedure as for batch gradient descent is then performed on the data by computing the mean gradient, and then updating the parameters based on this value [40].

2.4.4 Convolutional Neural Networks and common features

Convolutional neural networks (CNN) is a design type of neural networks that has at least one layer where it uses convolutions instead of general matrix multiplication. The idea was initially inspired by how the visual cortex of animals is organized. While the neural networks previously discussed are fully connected in the way that every neuron on one layer is connected to every neuron on the next layer, CNNs are not fully connected. Regular feed forward neural networks have shown to both be prone to overfitting, and to have the issue of being computationally heavy. CNNs instead tries a different approach by transforming the data to a format that is easier to process, without losing too many critical features of the data that is necessary in order to train the network [45].

A convolutional layer takes a matrix as input, processes this matrix using a kernel and outputs a new matrix. If we consider the case where the input is represented as a matrix with dimensions $n \times n$, and the kernel is a matrix with dimensions $k \times k$, the kernel is then shifted and multiplied with each portion of the input data. Given the dimensions,

the total amount of multiplications is depending on how many shifts the kernel needs to perform before the entire input is considered. The results of the matrix multiplications after this process is a new matrix with either reduced dimensionality (valid padding), where the matrix has the same dimensionality as the kernel, or a matrix which has remained or increased its dimensionality (same padding) [45].

In CNNs, pooling layers are very commonly used to serve the purpose of making the network less computationally heavy, and also to extract dominant features from the data. In the same way as in convolutional layers, pooling layers use a kernel to shift over the data and perform some operation on the different portions. The result of this is as earlier, a new matrix with new dimensions, but of lower dimensionality in order to lower the computational cost [54]. There are two main types of pooling called max pooling, and average pooling. Max pooling takes the largest value of each portion, while average pooling takes the average value of the elements in each portion. An additional useful property of max pooling is that it is suppressing noise in the data [45].

Fully connected layers are typically also used as they are good at learning high level features from the previous convolutional layers, but also because they are required to enable classification [45].

2.4.5 Regularization - Dropout and Batch Normalization

Neural networks are prone to overfitting, and require techniques in order to prevent it. With unlimited computing power, the ideal way to combat this would be to average the predictions of all possible parameter settings [48]. While this might be possible for smaller networks, deep neural networks require less computationally expensive methods.

Dropout is a method which drops out units in a network with a manually selected probability, along with its incoming and outgoing edges. The dropout of a unit is temporary, and happens with a probability independent of the other units probability. The result of the process is a "thinned" out network. A network of size n, can be considered as having 2^n thinned out networks. During each step of training, a new thinned out network is sampled from the original network, and trained. It is still not practically possible to train all 2^n networks, but instead an approximate averaging method has proved to be effective. When training the

network, nodes are present with a probability p, constructing a thinned network from the set of 2^n possible thinned networks. When testing, instead of eliminating the node completely when picked, its outgoing weights are instead multiplied by p, and by that reduces the output values. This gives the network dropout two different behaviours decided by what task it is currently performing. Using this method of approximate averaging has shown to be very effective in making a network generalize well for a lot of classification problems [48].

Training deep neural networks is complicated by a phenomenon called internal covariate shift. This phenomenon occurs by a layer's inputs changing during training, as the parameters of the previous layer changes. This greatly slows down training, by requiring lower learning rates, especially if the data consists of nonlinearities [9]. Batch normalization addresses this problem by normalizing the layer inputs for each training batch. The batch normalizing transform algorithm is defined as follows [27]:

Input: Values of x over a mini-batch $B = \{x_1 ... x_m\}$ and parameters to be learned γ, β

Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

1. $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
2. $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
3. $\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
4. $y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$

Step 1: Calculate mini-batch mean. Step 2: Calculate mini-batch variance. Step 3: Normalize. Step 4: Scale and shift.

By using this transformation, we ensure that the network can continue to learn by having input distributions that have less internal covariate shift, and by that reduce the overall training time required. This is also due to it enabling higher learning rates without having the gradients explode or vanish. This transformation has also been shown to be regularizing the models, and in some cases removing or reducing the necessity of dropout [27].

Chapter 3

Experimental Study

3.1 Initial Discussion

Considering the properties of cryptographic protocols in combination with the features existing in Tor, we need some justification as to why it is realistic to construct any fingerprints at all.

Some considerations have been done during the implementation of the attack. A possibility is to train the classifier on the encrypted data and try to find patterns with that, but in order for encryption algorithms used by Tor to be secure, no information about the plaintext should be obtainable simply by looking at the ciphertext. The only information obtainable through the encrypted data is roughly the data size transmitted, but as discussed earlier Tor puts data into cells of a fixed size, which makes data from different servers indistinguishable in size. The encrypted data is because of this considered not usable for this attack. What is left considering then is simply the sequence of incoming and outgoing packets where the direction of packets is the only data upon which the classifier will try to build a fingerprint. The reason why this could bring promising results is based on the following key principle.

As pages may have big differences in the actual data it provides to the clients, it follows that the packet sequences are very unlikely to be identical. The times at which different tags such as images, styling and others are requested, will impact the packet sequence, and these differences between the webpages are what the classifier will try to construct fingerprints

from. Defensive measures proposed to combat these types of attacks, such as WTF-PAD and Walkie-Talkie, attempts to change the packet sequences enough such that classifiers will no longer be able to make these distinctions.

The attack will be performed on both undefended data, as well as data defended by Walkie-Talkie and WTF-PAD, in order to see if these methods are able to obfuscate the sequence in such a way that it can still be recovered by the intended receiver, without it being prone to fingerprinting.

As previous work has received some criticism on its approaches to construct a realistic scenario [29], it is important to create a setting which is as closely representing the real world as possible. To do so, the attack will be implemented in both a closed-world and open-world scenario.

3.1.1 Closed-world and Open-world scenario

In the closed-world scenario it is assumed that the user is only able to visit a fixed number of websites, which is known to the attacker. The attacker can then train his classifier on these known websites, and measure the accuracy of the attack by simply looking at the number of correct predictions from the total number of predictions made.

$$Accuracy = \frac{CorrectPredictions}{NumberOfPredictions}$$

The closed-world scenario has previously received criticism when discussing the feasibility of the attacks, but also because it is not believed to be mimicking reality in a good enough way. It can instead serve as a great way to improve the accuracy of the model, and then later test it in a more realistic scenario [29].

The open-world scenario is a more realistic approach when making assumptions about the real world. As it is unrealistic for an attacker to know the full set of possible websites, the open-world scenario instead creates a setting where the attacker chooses a smaller sample of websites to train on. We refer to the set of pages that the classifier trains on as the monitored set, while the remaining pages that the classifier has not been trained on are referred to as the

unmonitored set. When asked to make a prediction, the classifier produces prediction probabilities. This means that the classifier produces a vector consisting of probabilities, where the maximum probability in the vector represents the page which the classifier considers the highest chance of being the correct webpage. This results in the classifier predicting websites with different levels of confidence. In order to see how much the classifier's confidence over its prediction matters, the evaluation of the models will be performed for different thresholds in an attempt to find a specific threshold where the classifier might perform better. It is important to keep in mind that this approach could show cases where the prediction is correct, but the classifier shows very low confidence in its prediction [49].

3.1.2 Evaluation Method

The model's training accuracy will be determined for each epoch on the validation data for both the open-world and closed-world models. This will give an insight into how the performance of the models will improve as they are fed more training data. The closed-world accuracy will simply be calculated by performing a series of predictions on data that is never before seen by the model.

In the open-world scenario, the evaluation will be more detailed. As mentioned a prediction has to be at a certain level of confidence defined by a threshold. It is necessary to distinguish between unmonitored and monitored data. In a real world use-case, the model will be exposed to traffic which it will not have trained on. If the model is asked to make a prediction on this data, it would classify it as one of the web pages from the monitored set, resulting in a wrong classification. To avoid this, the model will provide a vector of probabilities, with the predicted page being the one with highest probability. Instead of straight away considering this prediction to be complete, we require the model's confidence to be at a certain threshold defined by a probability. The model will attempt classifying at different thresholds ranging from just 10% to 99% in order to find which confidence is required from the model to achieve its different performance levels. Furthermore, each prediction will be classified as a true positive (TP), true negative (TN), false positive (FP), or false negative (FN). A TP prediction is when the classifier correctly classifies a monitored page with a high enough level of confidence. TN predictions is when the classifier correctly classifies a page as unmonitored, meaning it did not show a high enough confidence towards any of the classes. A FP prediction is when the classifier mistakenly classifies an unmonitored page as

a monitored page. Lastly we have the case of a FN prediction where the classifier fails to classify a monitored page, or is not confident enough in its prediction of a monitored page.

With the described values, we can have a deeper evaluation of the performance, and also introduce precision and recall as a method of measurement. Precision is calculated by the following formula [19]:

$$Precision = \frac{TP}{TP + FP}$$

With the precision formula we can have a measurement for what proportion of the positive predictions were actually correct. To evaluate what the proportion of the actual positives were actually correct, we use recall, often referred to as the true positive rate (TPR), which is calculated by the formula [19]:

$$Recall = \frac{TP}{TP + FN}$$

When evaluating the effectiveness, we examine both precision and recall together. The reason for this is that they usually oppose one another where if one increases, the other decreases, and vice versa. As we also wish to evaluate the model for different thresholds, we can use ROC curves, (receiver operating characteristic curve), which plot a curve from the true positive rate (TPR) or the recall, and the false positive rate (FPR) defined as [20]:

$$FPR = \frac{FP}{FP + TN}$$

ROC curves are typically used in open-world scenarios where TPRs and FPRs are obtained for different thresholds similar to this case. By then plotting the TPR on the y-axis, and FPR on the x-axis, a point towards the top left corner is considered good if the goal is to simultaneously have high TPR and low FPR.

3.2 Tools

3.2.1 Development Environment

When making environment and design choices, the main focus throughout the implementation is keeping it simple, but effective. Python has become the language of choice for machine learning much due to the different libraries available as well as its simplicity. Python does however itself not compete well against other languages when it comes to computational speed, which means that computationally heavy parts have to be done through packages implemented in other languages such as C or C++. This makes Python with its packages written in C/C++, and more specifically version Python 3.8 the programming language of choice going forward [26].

For machine learning, TensorFlow provides a wide range of possibilities for neural networks [18]. As the task of improving the model is very important, it is crucial to get the level of feedback during training that TensorFlow provides. Other libraries do not necessarily provide the level of detail that TensorFlow does, which can make it difficult to get proper insight into the model's performance at different stages. Additionally, it is important to have complete control over the design of the network layers as we want it to be optimal for the type of data that will be fed to it. TensorFlow was initially a low level library which provides these features, but more recently the Keras library can be used at a higher level of abstraction in order to make some steps easier [1]. The main advantage of the new Keras library is how it allows for easy implementation and simple code, while still optionally returning advanced diagnostics when asked for it through the underlying TensorFlow resources. In the implementation the TensorFlow version used is TensorFlow 2.1.

3.2.2 Hardware setup and utilization

As training deep neural networks is computationally heavy, some considerations need to be made. A deeper and more advanced neural network results in more calculations, especially through matrix multiplication. As matrix multiplication easily can be parallelized, graphic processing units (GPUs) have shown to be much more effective in training neural networks in comparison to central processing units (CPUs) [46]. This means that if we can run the program on a powerful GPU, it will give possibilities for a much more complex model than if training is performed on a CPU. This gives two options. One is to run the program locally on a computer with a GPU, and the other is to rent computing power through the cloud. While typically renting the computing power is cheaper than buying a GPU for personal use, I still wish to make an attempt with my personal Nvidia Geforce GTX 1070 GPU which my own personal computer has set up.

Forcing the program to utilize the GPU requires some software installations. The CUDA toolkit provided by Nvidia provides a development environment which enables GPU-acceleration in high performance programs [37]. The library is built in C/C++ which further improves runtime as well as the ability to easily utilize it through other languages such as Python. What CUDA enables is firstly Cuda blocks, which is a collection of threads. Memory is then enabled to be shared within a block among all threads. Lastly, Cuda enables threads to wait until all the other threads have reached a particular point of execution, and by that lets a program easily parallelize without facing typical issues when parallelizing a program.

Nvidia also provides a deep neural network library named cuDNN which through the possibilities provided by CUDA presents powerful implementations for routines performed in neural networks such as backpropagation, forward propagation, pooling, and more [38]. Importantly it also enables matrix multiplication to be performed with a significantly less amount of memory, which typically could be a problem during training.

The hardware setup of the machine used to train the deep neural networks consists of:

1. NVIDIA Geforce GTX 1070 16GB (GPU)
2. Intel Core i5-7600K (CPU)
3. DDR4 16GB 2400Mhz (RAM)

3.3 Data Collection

Initially the plan was to gather data and build a new dataset. However, while looking at previous work done, there already exists a few datasets of the type necessary for this attack. By using the same techniques as in previous work for collecting data, there should be no reason for the attack to be performing any differently on a new or old dataset. Instead, it allows for further validation of the model by performing the attack on multiple datasets instead of only one. This will reduce the chance of there being any unforeseen anomalies in the data collection process which could give misleading results and time consumption.

3.3.1 TCPDUMP, Tor-Browser Crawler and picking URLs

When constructing the datasets it is important that it is performed in the most realistic way, while at the same time gathering large amounts of data. This makes it infeasible to do the browsing and packet tracing manually, and instead requires some tools to automate the process. The publicly available datasets used have all gathered the data in a similar fashion. Tor Browser Crawler allows for automated web crawling on Tor [2]. It is based on the Python libraries Selenium which allows for automated web browsing, and Stem which is a controller library for Tor [51]. By utilizing these libraries, the Tor browser crawler automates the task of browsing pages, and provides realistic transmission control packet (TCP) streams for further analysis.

While running the Tor browser crawler, the packet data needs to be picked up, stored and analyzed by another program. For this task TCPDUMP is typically used as it is both simple to use, and comes with any Linux based operating system. TCPDUMP can provide detailed data on ingoing and outgoing TCP streams, but in this thesis, we only focus on the directions of packets in a sequence of packets sent between a webserver and a user. This information can easily be obtained through sorting the traffic by its IP-addresses, and determine the direction a packet has by looking at the sending and receiving IP-address. As the traffic is from Tor, the packets coming from the webserver will display the IP-address of the first node. [4].

Since manually deciding which pages to fingerprint can be a time consuming process, the Alexa top list provides a wide range of popular websites. When constructing the datasets it is typically a subset from this top list that defines the pages to be used for the attack [7].

3.3.2 Dataset, Data format and Defense simulations

The Closed-world dataset is built from visiting 95 of the sites listed on the Alexa top 100 list. Each site was visited 1000 times by the Tor browser crawler, while dumping the packets with TCPDUMP. The sites were visited with specific methods in order to account for factors such as long-term short-term variance, by crawling each site in chunks in a round robin fashion [47]. By doing this, the data will account for variance of the site over time. Each data point in the training data is a vector of the packet trace from one of the 95 possible sites, while the actual label defining the site is stored in another vector which contains the correct labels of all the traces.

The open world dataset is constructed by visiting 40 716 sites from the Alexa Top 50 000 list, while excluding the first top 100 pages. Each site is visited only once. This is due to the fact that these pages will only serve as unmonitored sites in the open world setting, while the pages we wish to correctly classify require more data. Also, an important fact is that the size of the unmonitored set is orders of magnitude larger than the monitored set. This is to further mimic a realistic scenario where the majority of the traffic will be originating from unmonitored sites.

The data is sectioned into training data, validation data and test data. The training data will be used for training the model, and the validation data is used to estimate the training performance. When the model is finished training, its accuracy will be measured by running a series of predictions on the test set. Each of these sets consists of a X matrix and a Y vector, where elements in the X matrix is a vector representing the packet trace from a website. A packet trace is simply a series of the numbers 1, and -1, where 1 represents an outgoing packet, and -1 represents an incoming packet. Each trace is then made out of 5000 packets. By selecting a specific packet trace from X, we can check its site by looking at the corresponding i'th element from the Y vector. The Y vector consists of a series of numbers functioning as labels for the traces. An example from the data could be as follows:

$X_{train}[5] = [1, -1, ..., 1]$	*trace number 5 with length 5000 consisting of 1s and -1s*
$Y_{train}[5] = 6$	*trace number 5 labelled as page 6*

3.4 Model design and training

3.4.1 Picking a suitable model

Fingerprinting attacks on the Tor network is not a new topic, and has been attempted with varying results over the years, as mentioned in Section 1.4. In 2009, Hermann et al. [24] attempted producing fingerprints by using a Naïve Bayes classifier, which saw great results on protocols not designed to be anonymous. On Tor however, it only reached 2,96% accuracy. Subsequent research was able to greatly improve the accuracy on the dataset produced in [24]. Then, in 2014, Wang et al. [56] proposed the k-NN attack, which utilized the k-Nearest-Neighbour classifier on features such as packet ordering, incoming and outgoing bursts, and more. This slightly new approach achieved impressive results with a closed-world accuracy of 91%, and an open-world accuracy of 86%. Much more similar work has been done achieving roughly the same level of accuracy. As we have the possibility to gather and label large amounts of data, at the same time as computers have gained increased power, Sirinam et al. in 2018 proposed an improved attack by using deep neural networks [47].

When implementing the attack, there is large amounts of data already available from previous research projects. At the same time it is feasible to gather more should it be necessary. With this in mind, combined with the effectiveness of neural networks, the fingerprinting process will be performed by using deep neural networks. The model design will be based on the model proposed by Sirinam et al. The reason for using this specific neural network design is primarily as the scope for this thesis is not to dig deep in the machine learning topic, but rather use existing machine learning techniques and discuss its implication in the cyber security field.

3.4.2 Neural Network Design

The model design is visualized in figure 3.1, which displays the blocks as well as the parameters given to contents of each block.

The input data fed to the first block is the vectors representing different packet traces. These are of length 5000, meaning that each trace consists of the direction of 5000 different packets. Feeding the data in 1D is required before the fully connected layers, but might as well be performed at the start.

Block 1 has to accurately fit the input data. As we have -1 and 1 as possible input values, it is important to pick an activation function that captures negative inputs. For this the model uses the ELU, a close relative of the ReLU function [53], but in difference to ReLU, returns a value between 0 and -1 when fed numbers less than 0. By using this function the first block will successfully capture the incoming packets, allowing the use of different activation functions in the blocks going forward. Blocks 2-5 then instead uses the ReLU activation function which simply outputs a value between 0 and the input.

Each block has a Convolutional 1D layer with the same kernel size 1x8, but increased filter size as the network gets deeper. The filter size is the dimensionality of the output space, which in practice means the number of output filters in the convolution. The size of the convolutional layers increases through the network, but are kept at a level that does not make the training time too long. The stride in each convolutional layer is set to 1, meaning that it will iterate over all values of the input, and together with padding set to "same", produce outputs of the same dimensionality as the input by padding the ends if necessary.

Each block then performs batch normalization after both convolutional layers with the intent of making the network still learn efficiently. The activation layers then run the activation function ReLU on the nodes

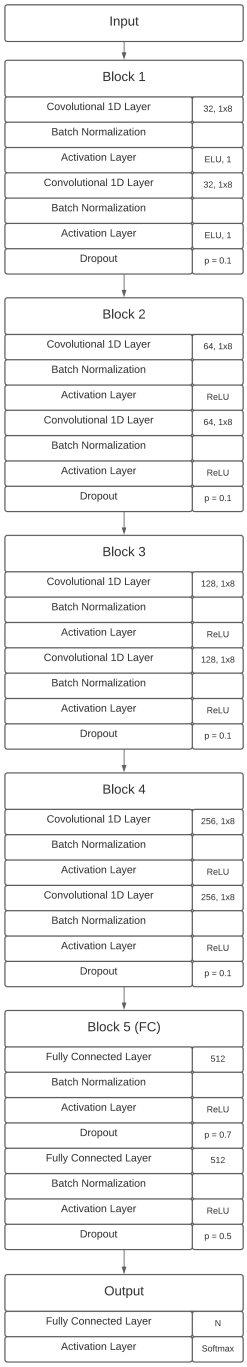


Figure 3.1: Deep neural network design

from the previous layer. Blocks 1-4 uses max pooling with the intent to reduce noise. This will then lead to less chance of overfitting, as well as lower computational cost, while still maintaining the necessary information in the network. Max pooling is performed with the same pool size and stride length in all blocks. The striding during max pooling is done in steps of length 4, reducing the computational time, resulting in a filter of size 1×8 which is padded to the correct size if necessary. Lastly, blocks 1 through 4 performs dropout with a rate of $p = 0.1$ before passing the values to the next block.

The 5th block is slightly different to the previous blocks. It consists of two fully connected layers and performs dropout at two occasions with much higher probability than in the previous blocks. While the goal of the previous blocks primarily is to capture the features in the data, the fully connected layers purpose is to classify those features. The fully connected layers tend to be more prone to overfitting [45], and in order to prevent this the level of dropout is increased. Finally, the softmax activation function [10] takes the output from the fully connected layers which classified the datas features. The reason for using softmax is simply because fingerprinting multiple webpages requires an activation function capable of multi-class classification, in contrast to ReLU.

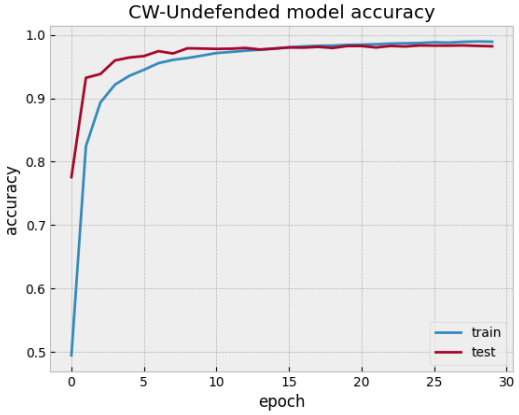
In summary the model starts off in block 1 with the ELU activation function to capture negative values. Through block 1-4 the model uses ReLU and convolutional layers to capture the features of the data. The output of the convolutional layers is then fed through fully connected layers to classify the features, before the output layer uses the Softmax activation function to perform multi-class classification. At all steps measures are made to improve computational speed as well as preventing overfitting.

3.5 Training the Neural Networks

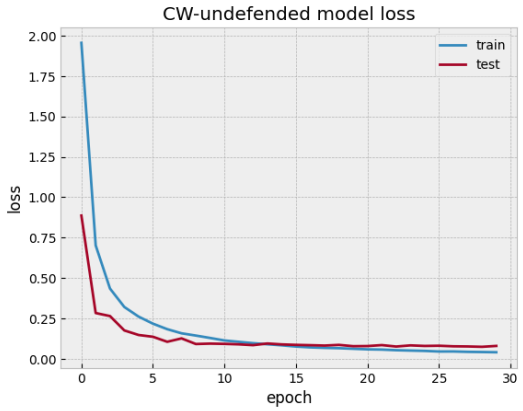
When evaluating the training performance of all the models, the training and validation accuracy will be considered together with the training and validation loss.

3.5.1 Training the closed-world neural networks

For the undefended model, we can see that the training accuracy is high both in validation and training already before the 10th epoch. Similarly, the loss drops quickly, further displaying the rapid learning of the model. While the accuracy does not seem to improve much over the last 10 epochs, it can still make quite a big difference if the model is able to improve just some fractions of a percent. At the end of epoch 30, the training of the model finalizes with a training accuracy of 0.981, a validation accuracy of 0.9819, a training loss of 0.0408, and a validation loss of 0.08.



(a) Undefended model accuracy during training and validation

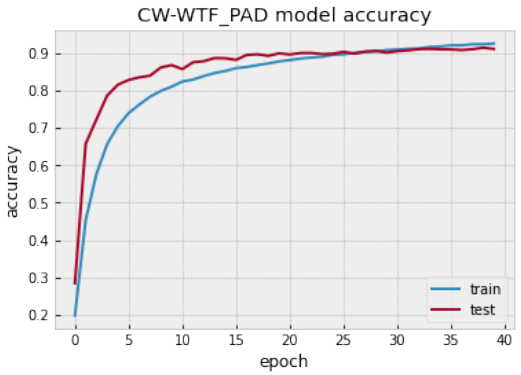


(b) Undefended model loss during training and validation

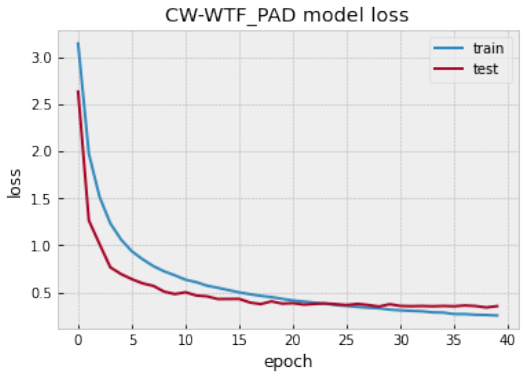
Figure 3.2: Plots of the closed-world undefended model performance during training

Next we look at the training of the WTF-PAD model. The first notable difference from the undefended model is that it requires more epochs. The model runs for an additional 10 epochs, resulting in a total of 40 epochs. This gives an indication that training the model on defended data is harder, as it should be. Similarly to the undefended model we can see how the validation curve represented by the red line is increasing at a slightly higher rate than

the training curve in blue. This could be due to a couple of reasons, with the more likely reason being that the network performs dropout. It could also mean that the validation data is generally less difficult to predict, but as the trend shows through all the models it strengthens the case of dropout. Additionally, this trend does ensure us that the model seems to not be overfitting the training data. The main difference from the undefended model is how the WTF-PAD model actually keeps improving its training accuracy all the way up to the 40th epoch. The reason for stopping at the 40th epoch is because at that point, the validation accuracy actually starts dropping. This means that with the current model design, it seems to start overfitting after epoch 40. It could also mean that it may be possible to further improve the accuracy by adding more methods to reduce overfitting, while at the same time increasing the number of epochs. At the end of training, the WTF-PAD model has a training accuracy of 0.9258, a validation accuracy of 0.9112, a training loss of 0.2538 and a validation loss of 0.3533.



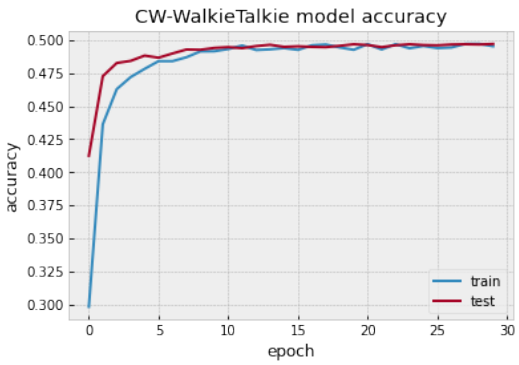
(a) WTF-PAD model accuracy during training and validation



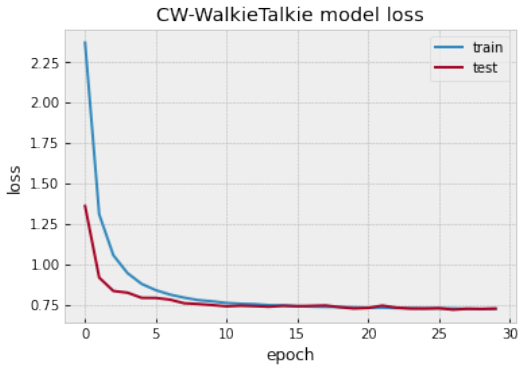
(b) WTF-PAD model loss during training and validation

Figure 3.3: Plots of the closed-world WTF-PAD model performance during training

Lastly for the closed-world training we look at the Walkie-Talkie model. Immediately we can see how the model learns very fast, in similar fashion to the undefended model. Already within the first 10 epochs, the accuracy is close to reaching its maximum. We can also clearly see how the accuracy is unable to surpass the theoretical maximum accuracy for Walkie-Talkie defended traces as it converges towards 0.5. The Walkie-Talkie is at its final stage able to reach a training accuracy of 0.4953, a validation accuracy of 0.4970, a training loss of 0.7246, and a validation loss of 0.7273.



(a) Walkie-Talkie model accuracy during training and validation

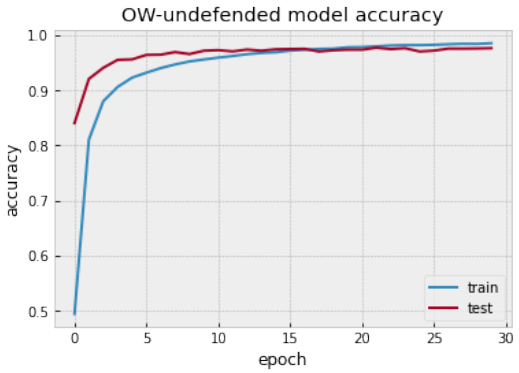


(b) Walkie-Talkie model loss during training and validation

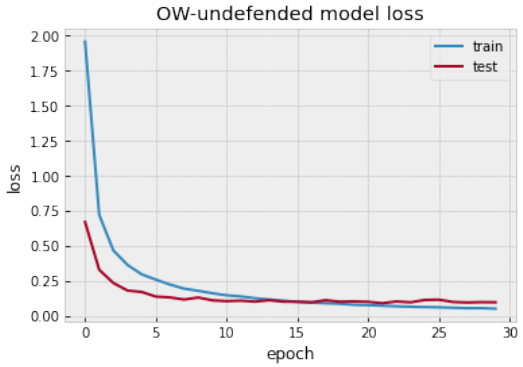
Figure 3.4: Plots of the closed-world Walkie-Talkie model performance during training

3.5.2 Training the open-world neural networks

In the open-world undefended scenario, we once again see that the model is able to quickly learn from the undefended data. The training of the model shows little difference in comparison to the closed-world undefended model, and finalizes its training with a training accuracy of 0.9856, training loss of 0.0540, a validation accuracy of 0.9767 and a validation loss of 0.0997.



(a) Undefended model accuracy during training and validation

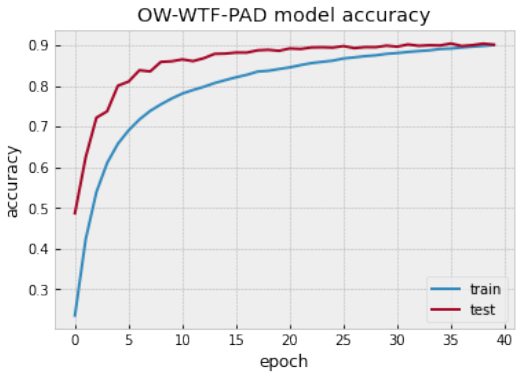


(b) Undefended model loss during training and validation

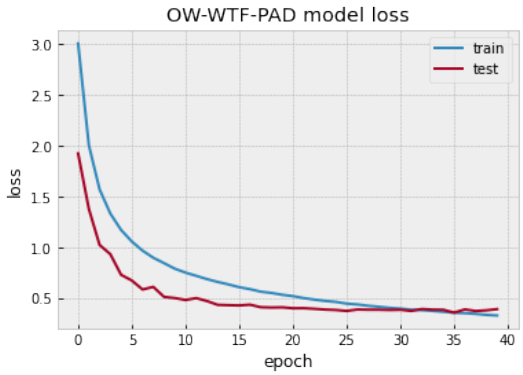
Figure 3.5: Plots of the open-world undefended model performance during training

For the training of the open-world WTF-PAD model, we again require 40 epochs similarly to the closed world scenario. While the training of this model is also very similar to the closed

world model, it does however show a slightly smaller gap between the training and validation loss in the later epochs. This could mean that the model handles overfitting slightly better, and might be either closer or further away from a theoretical maximum accuracy than the closed-world model.



(a) WTF-PAD model accuracy during training and validation

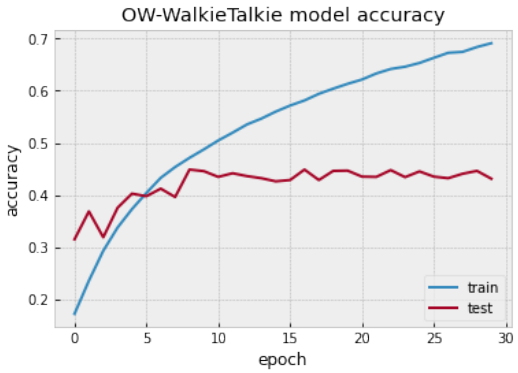


(b) WTF-PAD model loss during training and validation

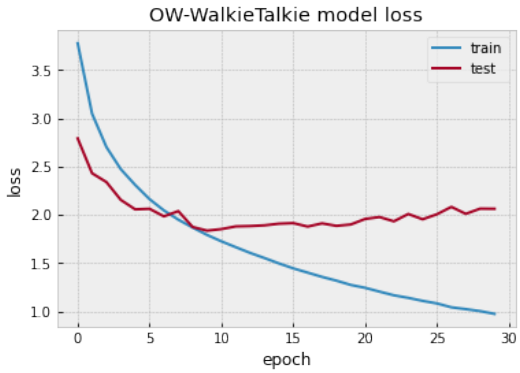
Figure 3.6: Plots of the open-world WTF-PAD model performance during training

Finally we have the last stage of model training with the open-world Walkie-Talkie model. In contrast to the previous models, we can immediately see how the results are strikingly different from the closed-world model. In the closed-world model we saw how it was unable to surpass 50% accuracy as expected with the theoretical maximum limit for the Walkie-Talkie defense. In this open-world scenario, we can see the importance of proper evaluation of a model, as the training accuracy surpasses 50%. While it is hard to say with certainty what the cause of this might be, it is likely that this is a result of the model remembering its guesses between sensitive and non-sensitive pages, and by that causing it to falsely defeat the theoretical limit. This means the model is fooling itself, as is clearly shown by the validation accuracy's inability to follow the training accuracy. By comparing the validation accuracy to the closed-world validation accuracy, the more realistic open-world scenario seems to be more challenging, and the maximum accuracy is reached already quite early in the epochs without being able to improve significantly over the later epochs. The open-world validation accuracy and loss is also quite unstable. This could be as a result of several things, such as a too high learning rate that might lead to big changes from one batch to another, and by that making the network unlearn previous features. It could also be a result of the model not being complex enough, making it unable to fully capture all features from the data. As the hardware setup has its limitations, the model performs quite well in the current environment.

At the end of training, the open-world Walkie-Talkie model achieves a training accuracy of 0.6908 and a training loss of 0.9753, a validation accuracy of 0.4315 and a validation loss of 2.0625, with the importance in this case located in the validation results.



(a) Walkie-Talkie model accuracy during training and validation



(b) Walkie-Talkie model loss during training and validation

Figure 3.7: Plots of the open-world Walkie-Talkie model performance during training

In summary, the open-world training shows small differences from the closed-world training for the undefended model and the WTF-PAD model. In both cases, accuracies very similar to the closed-world models were achieved. The open-world Walkie-Talkie model did in contrast to the other models display big differences as the training accuracy heavily differed from the validation accuracy, and validation accuracy ending lower than for the closed-world model. The training time of the models were also similar to the closed-world models, with an average of 105 seconds for each epoch.

3.6 Model Evaluations

In order to find the final performance level of the various models, a final test set will be used. This set consists of data never before seen by the networks, and by running a series of predictions on this data, we will see how well the model is able to generalize.

3.6.1 Closed-World Evaluation

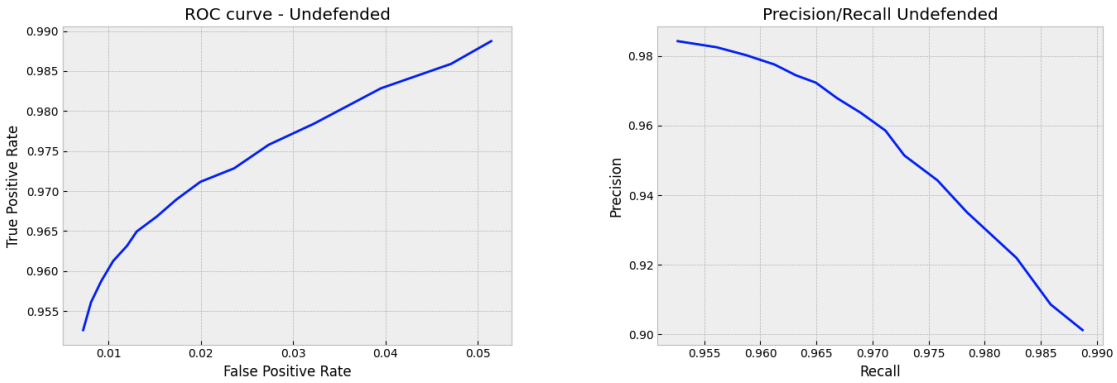
For the closed world evaluation, each model is asked to perform a total of 9500 predictions where each prediction is compared to a label to see if the prediction is correct. The accuracy will then be the number of correct predictions divided by the total number of predictions. Producing these predictions on undefended data, WTF-PAD defended data and Walkie-Talkie defended data, leads to a final closed-world accuracy of:

Undefended model accuracy:	0.9829
WTF-PAD model accuracy:	0.9112
Walkie-Talkie model accuracy:	0.497

As we can see, the model's performance on completely unseen data is very similar to the accuracy seen during training. This means that the model has successfully resisted overfitting, and is able to achieve high accuracies in the closed-world setting. While it might not be results that represent the real world as accurately as in the open-world scenario, it still provides useful insight into the models performance from training to testing.

3.6.2 Open-World Evaluation

For the open world evaluation, the models will in a similar fashion to the closed-world scenario be fed data from test sets, but the test sets will now also include unmonitored data. Each model will be asked to make a total of 29 500 predictions, where each prediction will count as a true positive, true negative, false positive or false negative. As this evaluation will be more detailed, the results may be interpreted in different ways depending on what is considered optimal. In a real world use case, it might vary what is most important to minimize or maximize. For example in criminal cases, it would not be acceptable to have a high rate of false positives, while in other cases having a high false positive rate might not be as damaging as long as the true positive rate is high.



(a) Undefended model ROC curve

(b) Undefended model Precision/Recall curve

Figure 3.8: Plots of the open-world undefended model ROC and Precision/Recall curves

We start off with the undefended model, and from the plots we see the ROC curves [20] and precision/recall curves [19]. By looking at the values on the axises, it is clear that the model has a very low false positive rate, while at the same time having a high true positive rate, and similarly has both a high precision and recall simultaneously. The TPR ranges from 0.956 to 0.988, the FPR from 0.007 to 0.05, and a precision value ranging from 0.9 to 0.98. This shows that even in the open world scenario, the performance of the model against undefended data remains very good. Depending on the scenario, the model can be optimized to either a high TPR, low FPR or a combination of the two. The difference between the optimizations are not significant for the undefended model, meaning that the model seems to be showing high confidence in its predictions. Overall, this shows that the undefended

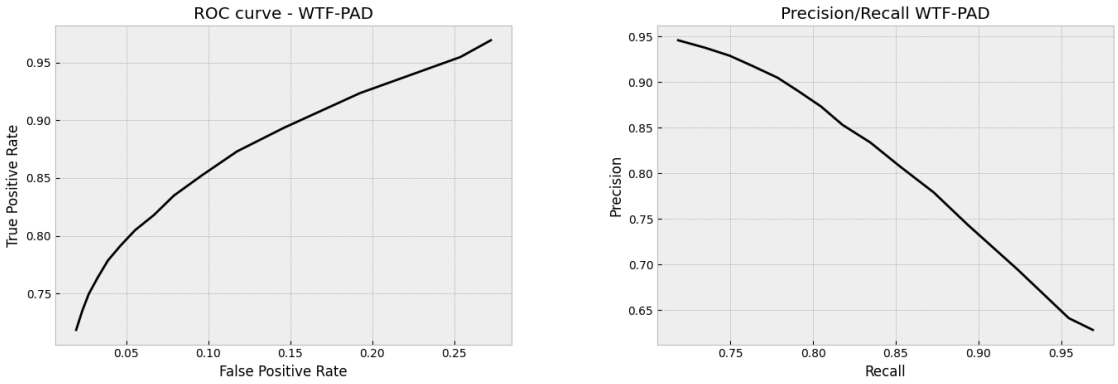
model did actually perform very well even in the open-world scenario, and shows results close to what was seen in the closed-world scenario.

Undefended Model Performance

Optimization:	Threshold	TPR	FPR	Precision
High TPR:	0.1087	0.9887	0.0514	0.9012
Low FPR	0.99	0.9526	0.0072	0.9842
TPR/FPR Tradeoff:	0.9502	0.9649	0.0130	0.9723

Table 3.1: *Table displaying the undefended model's performance for different optimizations*

For the open-world WTF-PAD model we can see that in comparison to the undefended model, the values on the axises are of more significance. Looking at the ROC curve, we see how the TPR ranges from 0.718 to 0.97, the FPR from 0.019 to 0.27, and the precision from 0.628 to 0.945. While they show a much larger range in values, they are still remarkably good considering that WTF-PAD aims to prevent fingerprinting. Similarly to the undefended model, different optimizations might be applied, and for this model seems to have a slightly larger effect. We see that the tradeoff between TPR and FPR is at a confidence level of 0.8207. This shows that while the model is still quite confident in its predictions, the WTF-PAD does have some effect on the difficulty of prediction.



(a) WTF-PAD model ROC curve

(b) WTF-PAD model Precision/Recall curve

Figure 3.9: Plots of the open-world WTF-PAD model ROC and Precision/Recall curves

WTF-PAD Model Performance

Optimization:	Threshold	TPR	FPR	Precision
High TPR:	0.1087	0.9692	0.2723	0.6283
Low FPR	0.99	0.7184	0.0195	0.9458
TPR/FPR Tradeoff:	0.8207	0.8524	0.0963	0.8077

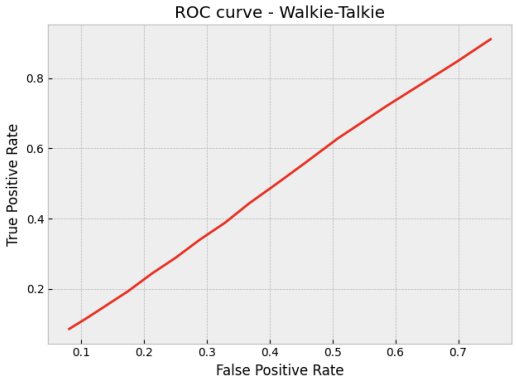
Table 3.2: *Table displaying the WTF-PAD model's performance for different optimizations*

At last we have the evaluation of the open-world Walkie-Talkie model. From the plots, we see that similarly to the results in training, this model shows strikingly different results to the WTF-PAD model and the undefended model. We also see that the TPR ranges from 0.11 to 0.91, the FPR from 0.08 to 0.75, and the precision from 0.32 to 0.357. This means that the TPR and the FPR is heavily influenced by the threshold rate. Perhaps the most unique result from this model is how, in contrast to the other two models, the precision/recall curve does not trend in the same direction for all thresholds. The highest precision rate is actually achieved at a confidence threshold of 0.659. This makes it possible to also optimize for a high precision rate. Additionally the model seems to have a very strict tradeoff between TPR and FPR, with the tradeoff optimization actually being a confidence threshold of 0.108, which is similar to the high TPR optimization. This can be justified by the fact that Walkie-Talkie should, in theory, never give the possibility of a prediction being very confident, as guessing between the dummy page and the actual page has to be done. This shows that Walkie-Talkie does significantly impact the performance in comparison to the two other models, but does not provide a complete protection against fingerprinting. It does seem like the security provided by Walkie-Talkie primarily arises from forcing the attacker to guess between the dummy page and the real page. This can be justified by the difference seen in this model from the other two, as well as the TPR and FPR tradeoff optimization threshold.

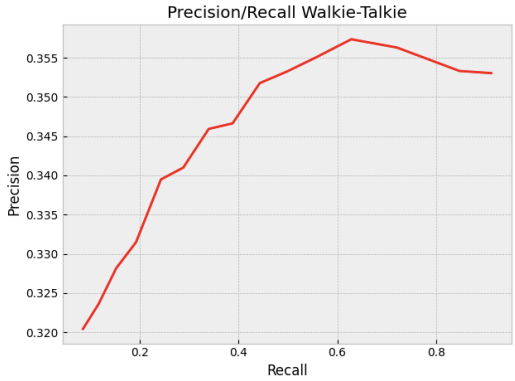
Walkie-Talkie Model Performance

Optimization:	Threshold	TPR	FPR	Precision
High TPR:	0.1087	0.9111	0.7513	0.3530
Low FPR	0.99	0.0855	0.0816	0.3204
TPR/FPR Tradeoff:	0.1087	0.9111	0.7513	0.3530
High Precision	0.6594	0.6282	0.5084	0.3573

Table 3.3: *Table displaying the Walkie-Talkie model's performance for different optimizations*



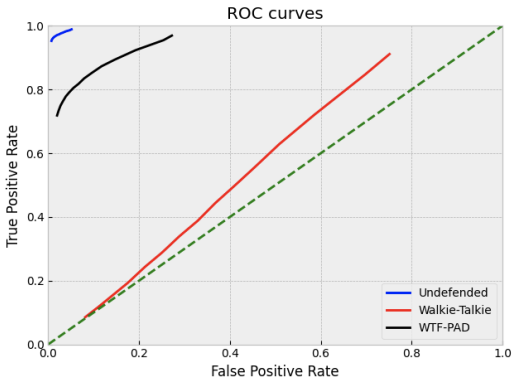
(a) Walkie-Talkie model ROC curve



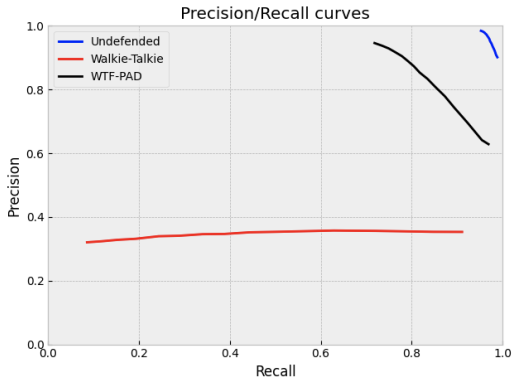
(b) Walkie-Talkie model Precision/Recall curve

Figure 3.10: Plots of the open-world Walkie-Talkie model ROC and Precision/Recall curves

By plotting the graphs of the different models together, we can get a clearer view of how they perform in comparison to each other. We can see that the undefended model model, and the WTF-PAD models curves are of quite similar looks, but with the undefended performance being higher than the WTF-PAD performance. The Walkie-Talkie curves show a big difference to the other two, by having both overall lower performance as well as different characteristics.



(a) ROC curves



(b) Precision/Recall curves

Figure 3.11: Plots of all the models ROC curves and Precision/Recall curves for comparison. The dotted line on the ROC curve plot displays the boundary a point must be above in order to not be purely guessing.

3.7 Discussion

It is debatable how effective this attack on the Tor network, considered in this thesis, might be in a real world scenario, and its true potential will likely only be known if some agency were to use it. However, one could argue that such an attack clearly shows how completely anonymous communications are difficult to facilitate. Additionally, the attack demonstrates that access to metadata alone can possibly compromise systems, and how modern techniques such as deep learning have the potential to achieve good results even while being limited to metadata. This does raise some topics which require some discussions.

3.7.1 Feasibility of Fingerprinting Attack

The type of Tor communications vulnerable to digital fingerprinting, is limited to specific types of web browsing. It is unlikely that other services such as chat, email, streaming, and more can be affected by this, as the contents of such traffic is very dynamic in comparison to web pages. This also implies that dynamic web pages should be more resistant to fingerprinting than a completely static web page. The attacker can then attempt to combat dynamic web pages by training models more often, and exposing the model to fresh data continuously. However, the issue with that is how web browsing has increasingly become a service tailored to clients specific preferences with the use of cookies and more. An example of this is how stores on the Internet often displays different material based on a users previous behaviour, making the page display different data to different users. This together with additional factors that affect packet ordering, would likely influence the attack's effectiveness.

With this in mind, it follows that the attack should be more effective if packet sequences are more static with regards to changes over time, but what if the clients actively try to disrupt the typical patterns? For example, instead of visiting the frontpage of a site, the client instead goes directly to the different directories of the web page. This means that unless the model has trained on different directories of a page, the client's anonymity should not be compromised. The overall complexity of the attack would then increase by the number of directories the web page has. These factors are all worthy of consideration, as the attack has only been done in an artificial environment. Even with measures taken for the environment to be as realistic as possible, there could still be factors unaccounted for in the real world.

There is also no reason to believe that traffic from a webserver hosted as a hidden service would be protected against this attack. This is due to the fact that using hidden services only makes the setup phase of circuits stricter, without adding any extra security measures against fingerprinting. Hidden services remain able to facilitate secret communications securely, by not publishing the necessary information needed to connect to it, or by only allowing a specific set of authorized users to connect. This means that if a group of users secretly share this information between each other, a third party would not know the page even exists, thus excluding possible fingerprinting attacks. Additionally, an attacker would never be able to train a model from the website's traffic if the server does not wish to connect.

The effects on VPNs is also worthy of a short consideration, as VPNs at a varying degree depending on the provider, advertises the possibility of anonymous communication. Constructing digital fingerprints on VPN traffic without extra measures implemented to prevent fingerprinting, should be easier than in the case of Tor. This is due to the fact that communications on the Internet is not performed by putting data into fixed sized cells in the way Tor does. When training a classifier, additional factors such as individual packet size, could be used to further strengthen the attack performance. From this it should also be reasonable to believe that if VPNs wish to add defense measures, they would have to be even stronger than the ones proposed for Tor, as the undefended traffic is even more prone to attacks. This means that while it still might be debatable to what degree this attack has an effect on Tor, there is a higher degree of certainty that such attacks can possibly compromise some of the services provided by VPNs.

Another topic of interest is if it would be possible to combine several attacks and through that enhance each attack's success rate. Correlation attacks have previously shown to be effective in different ways [8, 28]. Having control over varying amounts of nodes has shown to be a type of attack which can achieve results, and most likely already have in past cybercrime cases. This does raise the question whether a correlation attack can be combined with a fingerprinting attack, and possibly enhance a controlled node's chances of recognizing either a page, or a user, at different nodes simultaneously during transmission.

3.7.2 A critical view at attack performance

By looking at the results from the undefended models, it is clear that they both show very high performance. In the closed-world scenario we saw an accuracy of 0.9829, meaning

the classifier correctly identified nearly all pages given. Additionally, the belief in the results was further strengthened when the open-world model reached TPR values between 0.956 and 0.988, and FPR values between 0.007 and 0.05. This shows clearly how the packet ordering patterns in Tor traffic can be used to potentially deanonymize users, and why attention to the challenge is required. Important to keep in mind is also how the undefended model's performances are actually the ones representing the real world, as Tor currently employs no measures in order to break the packet ordering patterns and prevent webpage fingerprinting.

We have also seen how the WTF-PAD and Walkie-Talkie defense measures to varying degrees attempt to increase security. For these models, there is also some additional value in considering the closed-world accuracies. WTF-PAD seemed to improve the defence to some extent. The model required an additional 10 epochs, and fell a bit short of the undefended model's accuracy. If the case then is that the WTF-PAD only reduces the effectiveness of the attack by around 10%, it is clearly not effective. It is hard to tell exactly what the reason for this might be, as interpreting a neural network is difficult, and we can only assume what is likely to be the reason. As WTF-PAD attempts to add packets in a clever way, we can only assume that the usual pattern of packets is not sufficiently obfuscated. As dummy packets are sent based on the histograms gathered from a large number of pages, it is unlikely that the histograms are the cause. Instead a more likely cause is the rate at which dummy packets are sent. Sending dummy packets at a higher rate might reduce the attack performance, but does so unfortunately by increasing the transmission time.

Walkie-Talkie defended data seems to be the best performing defense of the two, but not without its catches. The closed-world model managed to get very close to the theoretical max of 50%, meaning that while the accuracy is lower than for WTF-PAD, its security almost purely comes from the guessing between the actual page and the decoy page. This could actually be a little problematic. If we instead ask the model to produce a Top - 2 prediction, meaning that it should return the most likely, and second most likely page, it could successfully identify both the actual page and the decoy page. This means that an attack can be improved to more than 50% accuracy, if some prior knowledge of the client exists, effectively increasing the theoretical max performance. For example, it is unlikely that a user only speaking English would browse a Spanish page. Walkie-Talkie is because of this relying heavily on clever picking of decoy pages. This means that it does to some extent prevent mass-deanonymization, as in order for the Top - 2 prediction to work, prior knowledge of each client is required. A good case for Walkie-Talkie is how the model is

quite unconfident in its predictions. This means that while the model in theory reaches a decent performance, in practical real-world use cases it might not live up to the confidence level required in specific situations such as courts. Considering these results, it seems that Walkie-Talkie does provide a certain degree of security, but does not defeat the attack. It is also possibly vulnerable to Top - N based attacks, meaning that if specific users with prior knowledge are targeted, the theoretical maximum accuracy of 50% possibly no longer applies.

3.7.3 Data Storage

As digital fingerprinting attacks heavily rely on access to large amounts of data, the topic of data storage is of interest. Currently as of May, 2021, the Norwegian government is working towards enforcing a new set of laws for digital surveillance [15]. While this thesis will not argue the ethics around such a topic, it should still be in everyone's interest to discuss, make decisions and enforce policies with informed opinions. The new laws on surveillance were partly implemented the 1st of January 2021, with the exceptions chapters 7 and 8, due to two ongoing trials in the European Court of Human Rights, as well as past cases in the British, French and Belgian court systems [42]. §7.7 in The Intelligence Services Act would allow for mass storage of border-crossing metadata by the Norwegian Intelligence Agency, for up to 18 months [43]. The proposal is both welcomed and resisted by different political parties [25]. As this type of data storage is exactly the type required to perform the fingerprinting attacks discussed in this thesis, it should be considered when discussions around data storage are made. It is important to be aware that even with heavily encrypted channels such as Tor, mass storage of metadata can threaten the anonymity of the users. Privacy considerations and interests of non-criminal users of the services provided by Tor, should be a part of such policy. An example of such a non-criminal entity could be whistleblowers. When a government has the possibility to enforce policy that influences anonymity, it could potentially make it harder for someone living in that country to speak up anonymously against wrongdoings made by the same country.

3.7.4 Difficulty of anonymity

After considering the technology provided by Tor, even with fingerprinting defenses, it is clear that creating an anonymous environment on the Internet is surprisingly hard. Systems

like Tor, that only a few years ago were believed to be completely secure even by the NSA, are now shown to have weaknesses. There are multiple reasons why achieving anonymity is hard. As Tor already faces challenges on topics such as speed, a solution adding too much extra traffic would simply not be viable. Naive solutions, such as transmitting all the data at once, would simply slow down the communications to a point where it is not usable anymore. This means that the type of defenses such as Walkie-Talkie and WTF-PAD considered in this thesis, are of interest, as they both are the type of defenses that do not slow down traffic too much. The question then becomes how can you add to the traffic, or modify the traffic, to such a degree that it is unrecognizable to an attacker by being indistinguishable from other webpages, but still remains recoverable by the receiver?

At its most basic, one could say that there are two main ways to facilitate completely anonymous web surfing. The first way is to have all pages appear similar. This would mean that a packet trace from two different websites should be so similar, that it would not be possible to distinguish them from each other. This opens for some naive approaches, such as padding packets in order to create similar sized packet sequences, and then sending all packets at once. The main problem with this approach is how vastly different pages should appear similar. In a worst case scenario, a solution like this would mean that a small sized page would have to pad packets to the length of a much larger sized page, and by that making communications too slow on a network already facing difficulties with speed.

The second approach would be to have all pages appear completely random. This means that visiting the same page at different times, would result in packet traces having little to no similarities. An approach like this is more realistic to achieve, but at the same time difficult, as seen with WTF-PAD and Walkie-Talkie. When browsing a page, the sequence of packets depends on the content at which the page displays, and at which times the content is required by the client. Simply randomizing this pattern would result in the page not being recoverable by the client, meaning that while the packet sequence has to appear random, in fact it can not be random to the client. The answer may then lie in solutions where a client and a server agree on a certain pattern enabled by public key cryptography. A solution like this might add some extra time to the setup phase, but then perform equally to other solutions during the transmission. As the pattern is agreed upon, and unique for each visit, packet sequences would to an attacker be indistinguishable from one another even if they all originate from the same page. It is important to note that solutions like this are purely theoretical. In order to actually implement such a defense mechanism, changes need

to be made at many steps, and especially the way a browser communicates with a server. Additionally, agreeing on a pattern might not be as simple as it sounds. After all, the pattern that currently exists in traffic is there for a reason. Agreeing on a pattern would imply that the client and server agree at which time for example a specific image should be loaded. This could then run into the same issues as the more naives approaches where the method simply adds too much network congestion. Facilitating completely anonymous communications is most definitely hard, and has already come a long way with Tor, but to have a completely anonymous channel is a challenge that requires more work.

Chapter 4

Conclusion and Future Work

The main goal of this thesis was to revisit the attacks on Tor proposed through recent research, and do a deeper dive into how such an attack is performed, what the performance of such an attack is, as well as look into defense mechanisms proposed. By doing research into these topics, new potential future research has occurred, and might require future attention.

By implementing modern techniques available in the field of deep learning, the models were able to find patterns in packets obtained from Tor traffic, leading to high rates of successful web page classification. The models were also separately trained on traffic defended by WTF-PAD and Walkie-Talkie, and showed decent results even on defended traffic. In conclusion, this research highlights the challenges in facilitating completely secure and anonymous communications. Through the improvements made in the field of machine learning, methods previously believed to be sufficient, are now shown to be vulnerable to a type of attack which through the existence of Tor has received little attention.

As previously discussed in Chapter 3, there are several topics with potential for future work. We have seen how the defensive measures currently provided do not seem to meet the level of security they both aim for. This opens possibilities for further research in website fingerprinting defense mechanisms, and possibly, if public key cryptography can be used to pre-determine a specific packet sequence. In the thesis the attack was exclusively performed on traffic from Tor. While Tor has a large number of users, a potentially larger user base is VPNs, with the services they aim to provide. If VPN routed traffic on the Internet does not contain any level, or perhaps only some level of fingerprinting defenses,

performing the attack on services claiming to provide anonymity could be of interest. As discussed, correlation attacks have also shown to have provided results through controlling various amounts of nodes. Research into the possibilities of combining correlation attacks with fingerprinting attacks, in order to strengthen the correlation attacks ability to identify a user at multiple nodes, is a possibility worthy of more attention.

Bibliography

[1] Keras: the Python deep learning API. **URL:** https://keras.io/, 2015.

[2] Tor-Browser Crawler. **URL:** https://github.com/webfp/tor-browser-crawler, 2016.

[3] RFC8446 - The Transport Layer Security (TLS) Protocol Version 1.3. **URL:** https://datatracker.ietf.org/doc/html/rfc8446, 2018.

[4] TCPDUMP/LIBPCAP public repository. **URL:** https://www.tcpdump.org/, 12 2020.

[5] The FBI TOR Exploit. **URL:** https://resources.infosecinstitute.com/topic/fbi-tor-exploit/, 03 2021.

[6] M.S. Ajay. Introduction to Neural Networks — Towards Data Science. **URL:** https://towardsdatascience.com/introduction-to-artificial-neural-networks-ac338f4154e5, 05 2020.

[7] Amazon". Alexa - Top sites. **URL:** https://www.alexa.com/topsites, 2021.

[8] R. Brandom. FBI agents tracked Harvard bomb threats despite Tor. **URL:** https://www.theverge.com/2013/12/18/5224130/fbi-agents-tracked-harvard-bomb-threats-across-tor, 12 2013.

[9] J. Brownlee. A Gentle Introduction to Batch Normalization for Deep Neural Networks. **URL:** https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/, 12 2019.

[10] J. Brownlee. Softmax Activation Function with Python. **URL:** https://machinelearningmastery.com/softmax-activation-function-with-python/, 10 2020.

[11] CK Conger. Google is fighting Symantec over encrypting the internet. **URL:** https://techcrunch.com/2017/03/27/google-is-fighting-with-symantec-over-encrypting-the-internet/, 03 2019.

[12] W Diffie and M Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi: 10.1109/tit.1976.1055638.

[13] R Dingledine, N Mathewson, and P Syverson. Tor: The Second-Generation Onion Router. **URL:** https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.html, 2004.

[14] P Eckersley. How unique is your browser? **URL:** https://coveryourtracks.eff.org/static/browser-uniqueness.pdf, 2010.

[15] Forsvarsdepartementet". Lov om Etterretningstjenesten (etterretningstjenesteloven). **URL:** https://www.stortinget.no/no/Saker-og-publikasjoner/Saker/Sak/?p=79451, 2019.

[16] M.J. Garbade. Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences. **URL:** https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb, 09 2018.

[17] D Goldschlag, M Reed, and P Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999. doi: 10.1145/293411.293443.

[18] Google Brain Team". Effective TensorFlow 2 — TensorFlow Core. **URL:** https://www.tensorflow.org/guide/effective_tf2, 01 2021.

[19] Google Developer". Classification: Precision and Recall — Machine Learning Crash Course. **URL:** https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall, 02 2020.

[20] Google Developer". Classification: ROC Curve and AUC — Machine Learning Crash Course. **URL:** https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, 02 2020.

[21] S Grigorescu, B Trasnea, T Cocias, and G Macesanu. A Survey of Deep Learning Techniques for Autonomous Driving. *Journal of Field Robotics, Online ISSN:1556-4967, 2019*, 2019. **URL:** https://arxiv.org/abs/1910.07738.

[22] The Guardian. 'Tor Stinks' presentation NSA document. **URL:** https://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document, Oct. 2013.

[23] C. Hauk. Browser Fingerprinting: What Is It and What Should You Do About It? **URL:** https://pixelprivacy.com/resources/browser-fingerprinting/, May 2021.

[24] D Herrmann, R Wendolsky, and H Federrath. Website fingerprinting. *Proceedings of the 2009 ACM workshop on Cloud computing security - CCSW '09*, 2009. doi: 10.1145/1655008.1655013.

[25] J Iden, I Jordan, and S Øvretveit. Digitalt grenseforsvar - en trussel mot personvernet? - Magma. **URL:** https://www.magma.no/digitalt-grenseforsvar-en-trussel-mot-personvernet, 02 2020.

[26] U. India. Why Python is the most popular language used for Machine Learning. **URL:** https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77, 06 2018.

[27] S Ioffe and C Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 37:448–456, 2015. **URL:** https://dl.acm.org/doi/10.5555/3045118.3045167.

[28] A Johnson, C Wacek, R Jansen, M Sherr, and P Syverson. Users get routed. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, 2013. doi: 10.1145/2508859.2516651.

[29] M Juarez, S Afroz, G Acar, C Diaz, and R Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014. doi: 10.1145/2660267.2660368.

[30] 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 (ESORICS)*, pages 27–46, 2016. **URL:** https://www.researchgate.net/publication/285648574_WTF-PAD_Toward_an_Efficient_Website_Fingerprinting_Defense_for_Tor.

[31] L Kaelbling and A Moore. Reinforcement Learning: A Survey. *Journal of Arti*

cial Intelligence Research 4, pages 237–285, 1996. **URL:** https://arxiv.org/pdf/cs/9605103.pdf.

[32] S. Kostadinov. Understanding Backpropagation Algorithm - Towards Data Science. **URL:** https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd, 08 2019.

[33] Y Levine. Almost Everyone Involved in Developing Tor was (or is) Funded by the US Government. **URL:** https://pando.com/2014/07/16/tor-spooks/, 07 2014.

[34] J. Liang. An Introduction to Deep Learning - Towards Data Science. **URL:** https://towardsdatascience.com/an-introduction-to-deep-learning-af63448c122c, 10 2018.

[35] S. Mishra. Unsupervised Learning and Data Clustering - Towards Data Science. **URL:** https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a, 05 2017.

[36] T.M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1 edition, 1997.

[37] NVIDIA". CUDA Toolkit. **URL:** https://developer.nvidia.com/cuda-toolkit, .

[38] NVIDIA". Developer Guide :: NVIDIA Deep Learning cuDNN Documentation. **URL:** https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html, .

[39] C. Paar, J. Pelzl, and B. Preneel. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 1st ed. 2010 edition, 2009. doi: 10.1007/978-3-642-04101-3.

[40] S. Patrikar. Batch, Mini Batch & Stochastic Gradient Descent - Towards Data Science. **URL:** https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a, 10 2019.

[41] M Perry. Experimental Defense for Website Traffic Fingerprinting — Tor Blog. **URL:** https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting, 09 2011.

[42] Regjeringen". Utsetter ikrafttredelse av to kapitler i e-loven. **URL:** https://www.regjeringen.no/no/aktuelt/elovkap7og8/id2784463/, 11 2020.

[43] Regjeringen". Lov om Etterretningstjenesten (etterretningstjenesteloven). **URL:** https://lovdata.no/dokument/NL/lov/2020-06-19-77, 2020.

[44] RL Rivest, A Shamir, and L Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. doi: 10.1145/359340.359342.

[45] S. Saha. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. **URL:** https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 12 2018.

[46] G. Sharabok. Why Deep Learning Uses GPUs? - Towards Data Science. **URL:** https://towardsdatascience.com/why-deep-learning-uses-gpus-c61b399e93a0, 07 2020.

[47] P Sirinam, M Imani, M Juarez, and M Wright. Deep Fingerprinting. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1–16, 2018. doi: 10.1145/3243734.3243768.

[48] 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*, pages 1929–1958, 2014. **URL:** https://dl.acm.org/doi/10.5555/2627435.2670313.

[49] A Stolerman, R Overdorf, S Afroz, and R Greenstadt. Breaking the Closed-World Assumption in Stylometric Authorship Attribution. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 185–205, 2014. doi: 10.1007/978-3-662-44952-3\{_}13.

[50] G Takacs. What is browser fingerprinting? **URL:** https://bdtechtalks.com/2020/04/17/what-is-browser-fingerprinting/, 04 2020.

[51] Tor-project". Welcome to Stem! — Stem 1.8.0 documentation. **URL:** https://stem.torproject.org/, 12 2019.

[52] Tor-Project-Blog". Traffic correlation using netflows — Tor Blog. **URL:** https://blog.torproject.org/traffic-correlation-using-netflows?page=1, 11 2014.

[53] University of Applied Sciences Northwestern Switzerland". Neural Networks. **URL:** http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf, 2018.

[54] M Valueva, N Nagornov, P Lyakhov, G Valuev, and N Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020. doi: 10.1016/j.matcom.2020.04.031.

[55] T Wang and I Goldberg. Walkie-talkie: an efficient defense against passive website fingerprinting attacks. *SEC'17: Proceedings of the 26th USENIX Conference on Security Symposium*, 2017. **URL:** https://dl.acm.org/doi/10.5555/3241189.3241296.

[56] T Wang, C Xiang, C Nithyanand, R Johnson, and I Goldberg. Effective attacks and provable defenses for website fingerprinting attacks. *SEC'14: Proceedings of the 23rd USENIX conference on Security Symposium*, pages 143–157, 2014. **URL:** https://dl.acm.org/doi/10.5555/2671225.2671235.

Appendix A

Complete tables containing the open-world model's results

Open-World Undefended Model Results:

Threshold	TP	FP	TN	FN	TPR	FPR	Precision	Recall
0.108749	9393	1029	18971	107	0.988737	0.051450	0.901267	0.988737
0.353283	9366	942	19058	134	0.985895	0.047100	0.908615	0.985895
0.530724	9337	790	19210	163	0.982842	0.039500	0.921991	0.982842
0.659480	9295	645	19355	205	0.978421	0.032250	0.935111	0.978421
0.752909	9270	547	19453	230	0.975789	0.027350	0.944280	0.975789
0.820704	9242	472	19528	258	0.972842	0.023600	0.951410	0.972842
0.869897	9226	399	19601	274	0.971158	0.019950	0.958545	0.971158
0.905594	9205	347	19653	295	0.968947	0.017350	0.963673	0.968947
0.931496	9185	305	19695	315	0.966842	0.015250	0.967861	0.966842
0.950292	9167	261	19739	333	0.964947	0.013050	0.972317	0.964947
0.963930	9150	240	19760	350	0.963158	0.012000	0.974441	0.963158
0.973827	9132	210	19790	368	0.961263	0.010500	0.977521	0.961263
0.981008	9108	184	19816	392	0.958737	0.009200	0.980198	0.958737
0.986219	9083	162	19838	417	0.956105	0.008100	0.982477	0.956105
0.990000	9050	145	19855	450	0.952632	0.007250	0.984231	0.952632

Open-World WTF-PAD Model Results:

Threshold	TP	FP	TN	FN	TPR	FPR	Precision	Recall
0.108749	9208	5446	14554	292	0.969263	0.272300	0.628361	0.969263
0.353283	9070	5075	14925	430	0.954737	0.253750	0.641216	0.954737
0.530724	8773	3850	16150	727	0.923474	0.192500	0.695001	0.923474
0.659480	8489	2927	17073	1011	0.893579	0.146350	0.743605	0.893579
0.752909	8294	2353	17647	1206	0.873053	0.117650	0.778999	0.873053
0.820704	8098	1927	18073	1402	0.852421	0.096350	0.807781	0.852421
0.869897	7930	1582	18418	1570	0.834737	0.079100	0.833684	0.834737
0.905594	7769	1337	18663	1731	0.817789	0.066850	0.853174	0.817789
0.931496	7646	1110	18890	1854	0.804842	0.055500	0.873230	0.804842
0.950292	7517	930	19070	1983	0.791263	0.046500	0.889902	0.791263
0.963930	7396	778	19222	2104	0.778526	0.038900	0.904820	0.778526
0.973827	7256	654	19346	2244	0.763789	0.032700	0.917320	0.763789
0.981008	7121	545	19455	2379	0.749579	0.027250	0.928907	0.749579
0.986219	6981	465	19535	2519	0.734842	0.023250	0.937550	0.734842
0.990000	6825	391	19609	2675	0.718421	0.019550	0.945815	0.718421

Open-World Walkie-Talkie Model Results:

Threshold	TP	FP	TN	FN	TPR	FPR	Precision	Recall
0.108749	8200	15027	4973	800	0.911111	0.751350	0.353037	0.911111
0.353283	7623	13953	6047	1377	0.847000	0.697650	0.353309	0.847000
0.530724	6486	11718	8282	2514	0.720667	0.585900	0.356295	0.720667
0.659480	5654	10168	9832	3346	0.628222	0.508400	0.357351	0.628222
0.752909	5008	9098	10902	3992	0.556444	0.454900	0.355026	0.556444
0.820704	4480	8203	11797	4520	0.497778	0.410150	0.353229	0.497778
0.869897	3989	7351	12649	5011	0.443222	0.367550	0.351764	0.443222
0.905594	3493	6584	13416	5507	0.388111	0.329200	0.346631	0.388111
0.931496	3059	5784	14216	5941	0.339889	0.289200	0.345923	0.339889
0.950292	2596	5017	14983	6404	0.288444	0.250850	0.340996	0.288444
0.963930	2189	4259	15741	6811	0.243222	0.212950	0.339485	0.243222
0.973827	1736	3501	16499	7264	0.192889	0.175050	0.331487	0.192889
0.981008	1374	2813	17187	7626	0.152667	0.140650	0.328159	0.152667
0.986219	1056	2207	17793	7944	0.117333	0.110350	0.323629	0.117333
0.990000	770	1633	18367	8230	0.085556	0.081650	0.320433	0.085556

Appendix B

Links to source code and dataset

The source code used in the thesis is available at GitHub:
Link: https://github.com/HallyB96/TOR-Deep-Fingerprinting-Master-Thesis.

The dataset used to obtain the results is provided by Sirinam et al. [47] and can be downloaded from:
Link:
https://drive.google.com/drive/folders/1kxjqlaWWJbMW56E3kbXlttqcKJFkeCy6?usp=sharing.