A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Bachelor of Science honours

in

Computer Science

Final Year Project: Final Report

Name: [Haren Upadhayay]

Student ID: 001023099

School of Computing and Mathematical Sciences
Faculty of Liberal Arts and Sciences

University of Greenwich

Supervisor: Dr Atsushi Suzuki

Submission date: April 2022

Word count: 8426

Final Year Project:
Final Report

[Haren Upadhayay]

School of Computing and Mathematical Sciences
Faculty of Liberal Arts and Sciences

University of Greenwich

30 Park Row, Greenwich, United Kingdom

ABSTRACT. Several companies with an online presence such as with websites face many threats in the online world. One of these threats are what are called 'Botnets'. There are several ways of combating these attacks, however as attacks become more complex and more frequent it becomes harder to battle these threats manually. We can employ Artificially Intelligence to identify as well as block these malicious attackers for us more efficiently and accurately. Our methods include previously generated data compared collecting data our self. Previous work has used artificial intelligence This paper will explore several ways we can implement our AI including the network architecture, classification and others.

Keywords: Artificial intelligence, machine learning, Botnets, Security.

Preface

The main idea about this project is to employ artificial intelligence in order to identify botnet packets hidden in network traffic. We explore different types of machine learning techniques and explore which is appropriate for our purpose.

Acknowledgements

I’d like to thank my friends and family for getting me through my time at Greenwich

Table of Contents

List of Figures

Chapter 1

Introduction

Botnets are classed as a network of computers, which have been compromised and infected with malware with the intent of infecting even more computers and networks. They are all controlled by a malicious actor and its main task is to spread itself across as many systems as possible without the user's knowledge [15]. They are often used in DDoS (Distributed Denial-of-Service) attacks or even used to steal information and spread spam. Machine learning is one method to increase the accuracy of detecting botnets by taking a currently existing data set and then recognising patterns of anomalous data, thus being able to identify whether incoming traffic is part of a botnet or not. It will then take what it has learned and apply this in the real world and deny these botnets from entering a server, by using the patterns it has learned from the training data it has been supplied. In many cases, this can be more accurate than a human, and since it is an automated process, it can be much faster in identifying botnets. However, this is highly dependent on the implementation of the AI (Artificial Intelligence) as well as the data-set it is provided with.

1.1 Overview

In this section, we will discuss the background of information security and the threats botnets create to companies online. We will also discuss how machine learning can be utilised in order to combat these threats.

1.2 Background

In the "Top 10 Security Concerns for cloud-based services" blog, the author detail how "cloud computing" are under many different threats ranging from inside threats and outside threats. One

of the threats that cloud computing face is data breaches, as much of the sensitive data is being stored online rather than being on premise. Hijacking of accounts is also another threat cloud-based computing, to combat this we need to ensure that we have proper account password policies such as changing it regularly and ensuring that it is complex enough. Cloud-Based services are also under threat from malware injection, for this reason these cloud-based services need to ensure that they are monitoring what traffic is coming into the server and try and prevent any attacks that may happen to the server. DDoS attacks are also a serious threat as it aims to prevent legitimate users from being able to use your website and can sometimes be a case of a distraction from even more harmful attacks such as taking down a firewall and attempting to steal information [29].

The ”Internet Security Protection in Personal Sensitive Information,” Journal details many different threats any user can face, whether it discusses more “human” exploits such as phishing and more “technical” exploits such as brute force attacks, or in our case which we are interested in, botnets [26]. They go over how criminals use botnets to steal personal information by attacking larger companies who are more likely to have a database of customers information and can use these botnets to infect their systems and steal information. They then use their own systems as “hosts” and then use them to spread their bots to more systems to gain a hold of more information. These can be extremely dangerous as if they are not detected and stopped then it can result in them being spread in many systems and causing expensive costs in damage and ruin the reputation of the company if it were leaked that they have been compromised.

Botnets are growing to become a serious threat to the online world. These botnets are attacking in large numbers posing as legitimate users, to engage in criminal activities such as DDoS attacks, spreading spam, stealing information, or even infecting more computers to attack more servers. It is especially important for companies to defend them self against these types of attacks as not only could they result in financial damages as well as losing some public trust if a data breach were to occur.

Machine learning is the study of self-improving algorithms which use data and experience. It is part of the artificial intelligence field. These algorithms use what is called Training Data in order to make predictions or decisions without needing to manually program them to do so. Machine learning can be a powerful tool to do a variety of things from email filtering to predicting the stock market, or in this instance, using machine learning to be able to identify whether incoming traffic is a botnet or not.

In this project a machine learning algorithm will be developed that can tackle the use of botnets to attack a server. Preventing Botnets attacks on a server is a significant problem as online activities become more prevalent, so it is important to have proper protections against these attacks. While classic network analysis and identifying anomalous packets is a reliable way to identify botnet packets, as a server grows larger and accepts more traffic, this may be a harder and less effective

way of monitoring traffic. By applying machine learning to this problem, we can have a higher accuracy rate of identifying illegitimate packets at a much lower cost.

1.3 Problems

Trying to implement Machine Learning to botnet detection can be a challenging topic, however, a manual analysis of packets coming into a network can prove to be not very effective and it is by using machine learning and artificial intelligence the security industry has rapidly developed [6]. Since we are able to quantify variables of a packet incoming to a server, such as IP address, time taken, or even able to quantify the number of packets coming into a network in a certain time-frame, by using these variables as inputs for the machine learning model, we can then train the model accordingly

1.4 Aims

The overall aim of this project is to be able to utilise a machine learning algorithm to be able to identify botnet packet incoming to a system. This will be done with pre-generated traffic data. Ideally the algorithm should be able to identify atleast 90% of botnet from a set of network traffic using Deep Neural Networks (DNNs)

Chapter 2

Literature Review

2.1 Overview

In this section, we discuss the overall field of the project, this includes discussing the information security side of botnets, including the properties of them, as well as currently existing methods of combating them as well as the disadvantages of them. This section also will discuss various methods of the way Artificial Intelligence is employed in detecting these botnets.

2.2 Information Security and botnets

Analysing the DNS (Domain Name System), as well as any relevant attributes is one method of analysing network traffic to identify botnet packets, looking for abnormal behaviour, whether it be checking for high network latency or sudden spikes in traffic. These can be signs of bot activity. However, they do not necessarily mean there is a bot attack as there can be a reason for this increase in traffic. We can use the idea of analysing the attributes and using them as inputs of our machine learning model to teach it to identify patterns.

Monitoring the host computer is also another way to analyse whether it has been compromised, by monitoring the processes of the system and checking for any malicious or abnormal activity. While this is an effective way, it can be expensive as you will need to be constantly watching the systems processes to make sure nothing suspicious is happening. Furthermore, it could be a case of once you have detected it might be too late in a case of a botnet trying to steal information, in which case you would need to use methods to prevent the botnets from entering the server in the first place. We can then use the idea from this article that botnet traffic will typically hold different

attributes compared to normal traffic, we can then train the AI to learn what typical traffic looks like, and then if any packets are different, we can then flag it as potentially a botnet.

In the ”Cloud Threat Defence – A Threat Protection and Security Compliance Solution,” article the author details some ways to combat against online threats. One suggestion is through the analysis of logs. This can be things like the user access logs, NetFlow logs (which is the network traffic and the volume of it, and where it is coming from and the destination), security vendor logs/events and some other pieces of information. These logs are continuously analysed to identify any anomalous activities and then take any countermeasures to prevent any damage [3]. By using the data from these logs, we can potentially train a machine learning algorithm to be able to identify patterns which are found within botnet data, and block the traffic.

In the “Dynamic Defence Architecture for the Security of the Internet of Things” article, it discussed how traditional defence measures against hackers are becoming less and less viable as “IoT” devices become more common. To tackle this issue, a dynamic defence architecture for the security of IoT is proposed. It explains that It is made up of six security defence segments which include active defence, threat detection, security warning, security response, security recovery and defence assessment[14]. By using machine learning, we can fulfil two of the six segments, those being threat detection and security warning.

Businesses also face problems with IoT devices, as many of these devices are interconnected, however not all the devices are properly secured, S. Singh explains “stored data should be secured against theft, tampering & protected in the transit and at rest”[21]. Ensuring that this data is protected is vital as much of the data is sensitive and can cause great harm to the person if the data has been leaked.

N. Yildirim and A. Varol discuss threats that’s online and mobile banking can face, whether it be through phishing or more technical approaches such as SQL injection. It details some solutions against these whether it be using a One Time Password to prevent hackers from gaining access to accounts that do not belong to them, or more technical solutions as monitoring network coming in and out of the system. It expands upon the network sides of security and discusses how as more users start to use online banking, it can become much harder to monitor the traffic coming in and coming out. While there are currently existing tools that monitor network traffic, it can sometimes have a lower accuracy than desired. With this information it can be concluded that training a machine to identify botnets and other illegitimate packets can yield a much better detection rate and make the servers much more secure [28].

An article called ”Botnet Communication Patterns,” has more information on the ways botnets communicate with each other and using this information it can be used to help identify botnets. It goes into some of the topology of botnets. One topology is a “Centralized” type, this means that every bot is connected to a centralised C&C (Command & Control) server. This type is easy

to set up, has low latency, good scalability and has no special requirements in respect to protocol [25]. However, one of its greatest weakness is a single point of failure, in order to take down this topology, you just need to take down the centralized C&C server and all the other bots will be taken down. Another topology is a P2P (Peer to Peer), in this type every bot is connected to another bot. Furthermore, every bot has a potential to be a C&C server. In a botnet which is fully meshed, the latency between bots can be very low as commands do not need to be passed along more bots and can be given directly from one to another. Using this study, we can understand botnet behaviour and use this information and use it to help us in picking out the features we want to input to our machine learning algorithm.

Due to the nature of this type of topology, the number of connections can increase drastically with more bots, this will lead to having a maximum number of bots depending on the protocol used as operating systems will have a maximum number of Transport Control Protocol (TCP) Sockets. These sockets aid in being able to transfer information between two hosts [7]. Since the number of connections in a p2p topology is high, the visibility of the botnet increases and makes it much easier to discover, however simply taking down a few of the botnets will not be enough to stop it from working. Implementing P2P botnets are difficult as you need to find a way to find all the peers and a way to distribute commands reliably. One way of doing this is to hard code the list of peers and addresses, this can be very time consuming and take up a lot of resources, another way of doing this is by storing the initial list is to utilise cache servers, therefore the single point of failure being that initial list, assuming it has not been hard-coded. Furthermore, as the peers are interconnected, the whole topology can be compromised if one of the bots were taken over and sends commands to every other bot in the topology [10].

2.3 Artificial Intelligence

An AI that uses machine learning can be very useful for companies looking to protect themselves against attackers, since these botnets can cause a lot of damage it would be wise to protect your network from this. Other researchers can also use the way this paper has approached this topic and depending on the outcome, build upon what this paper has created or learn from any mistakes this paper may have made so they can come up with a more accurate solution.

One journal that has been explored had used SoNSTAR (Sonification of Networks for SiTuational AwaRness) which is a monitoring tool which is used alongside existing intrusion detection systems [5]. It reads every single packet entering and leaving a network and associates it with a sequence of packets. It then plays a sound depending on certain characteristics of said packet which then a human operator can see if there are attacks happening. Using this information, we can use the output of this and feed it through an AI and then let the AI decide what to do. So classing

characteristics of packets coming into a server, then using those characteristics as "inputs" for the AI to use might be an effective way to accurately identify whether the packet coming in is of a botnet or a legitimate user. Using the idea from this paper, we can conclude it is possible to train a machine learning algorithm to detect botnets.

Reinforcement Learning (RL) is one machine learning training method where we reward actions which we desire, and punish those action we want to deter. It emphasizes how to act based on the environment in order to maximize the expected benefits[20]. It does not require labelled data, however this does not mean that it will not use any training data. RL can be used in time-varying system such strategy games where users are playing against each other. This type of machine learning will not be entirely appropriate for our use case as we are focusing on input data where time has little affect on whether network traffic is a botnet or not.

Convolutional Neural Networks (CNNs) is one way of identifying bot nets within a network. One journal uses a machine learning algorithm which is effective in identifying P2P bots [4]. This system extracts flow-based features from packet headers and trains the detection model using CNNs This system uses a "decision tree" binary method to decide whether incoming packets belong to a botnet and uses CNN to train said AI. They have found this to be highly effective in identifying p2p botnets, using data from well-known p2p botnet data set. Deep Neural Networks (DNNs) are also another way of identifying botnets within a traffic. DNNs use Layers and an activation function to be able to get an output. Within these predefined layers, there may also be Hidden layers [17]. This type of machine learning algorithms can be very useful as we can input all the data that we need in the model and then customise the activation functions to get the best results.

In the "ABIS: A prototype of Android Botnet Identification System," Article, the authors explain how botnets can be found using an android device. The way they identify botnet is by "checking android applications" and seeing if they are malware or not. One way of identifying botnet within an application is via "static methods"[23]. One of these methods is by observing the API of the application as it is required to make sure the application can run with the operating system of the phone; another way is by looking at the permission that the application need and seeing if the application is asking for too much. There are also more "Dynamic Detection" techniques that can be used to identify botnets. One way of this is using honeypots, to lure in botnets and then capture their network traffic for further behavioural analysis. Another way of identifying botnets is through Traffic monitoring, this can be done through monitoring and analysing the DNS Traffic, however, as of late this may be of a less effective way of identifying botnets as they have been designed to generate less DNS queries. Furthermore, the process for DNS analysis is overly complex and does not show how the botnet was able to intrude the system and how it is spread. We can also use Behavioural Analysis detection to identify botnets. This technique focuses on the network traffic and observes the "pattern" of traffic and identifies any "abnormal patterns." With this, botnets can be separated in two forms: operational behaviour analysis and attach behaviour analysis. From

this article we can use the idea of using honeypots to collect training data for our machine learning algorithm, however for reasons discussed later in this paper, we will not be using this technique for data collection.

2.4 Summary

We can see that the use of botnets is a serious threat for any online company, and as technology advances, these types of attacks become more and more sophisticated. We can employ the use of machine learning algorithms in order to combat this threat and mitigate it as best as possible to protect ourselves.

From these articles we can identify that malicious traffic will typically have anomalous network traffic data, from this we can conclude that there will be a ”pattern”, which the machine learning algorithm can identify. Furthermore, since we know that botnets will ’infect’ other devices within a system, we can use the fact that certain devices will emit anomalous traffic when they are infected to identify when an attack has occurred.

Chapter 3

Requirements Specification

The final product should be able to use a currently existing data-set and identify bot-net packets from the training data provided, then it should. Ideally it should be able to identify these botnets faster and more accurately than a human operator.

3.1 Functional Requirements

This product must is to be able to process a data-set in such a way that a machine learning algorithm can then utilise in training and learn to identify botnets in a system. It should also print out the array sizes of the data-sets to ensure that it is fit for purpose.

For analysing the effectiveness of the machine learning algorithm, the model should be able to record different metrics then save this data into a variable so we are able to output training results in the form of a simple graph so a quick overview of how the training went can be analysed.

3.2 Non-functional Requirements

The machine learning algorithm will be able to train on a specific data-set as it has been tailored to specifically work with the machine learning algorithm. It should be able to correctly identify botnet data with at least 90% accuracy to consider it a successful model.

Its main limitation being that it will only be able to function on network traffic that has been modified in a specific way, furthermore, this model will be limited to classifying whether incoming traffic is a botnet or not, and not classifying the type of botnet incoming to a system.

Chapter 4

Design of the Proposed Methods / Systems

4.1 Network Architecture

When deciding upon the network architecture of our machine learning model, we must first consider the use case of the model, as well as the features of the options we have available to us

Convolutional neural network (CNN) is one type on network architecture that can be used for the machine learning algorithm. It consists of an input layer, a convolutional layer and an output layer. The convolutional layer takes in an input, transforms it in a specific way and then passes this information on to another layer. With these layers, you can use them to identify images. An example of this is the 'MNIST Handwritten digit classification' [11]. This uses CNN to convert the image into a a list of numbers and then passes this information through multiple hidden layers in order to classify the number that has been written. For this reason Convolutionals may not be suitable for this algorithm.

Recurrent Neural Network (RNN), use previous information in order to try and predict future events. Some of its uses include speech recognition, language translation and stock prediction.[19] RNNs take in sequential data and then tries to predict what comes next, it will loop the network until it reaches the end of the data, it then passes the final output to what is called a 'feed-forward layer' and with that we get the final output. An extension of this is called a 'Long Short-Term Memory' where it takes into context the entire data-set if needed, making it more advantageous compared to RNNs [2]. As the primary object is a classification task, not a prediction task, this type of Network architecture may not be suitable for this type of problem.

For the Machine learning model, a Deep Neural Network (DNN) was the choice of the model for the algorithm. One of the main advantages of DNN is that it does not rely too heavily on properly

labelled data. As well labelled data can be very expensive, and due to my lack of budget, DNNs are the cheapest way to get good results, although labelling might not be necessary, an effort has been made to still find data that is labelled properly to ensure the best results. [27] Another advantage of DNNs is that feature engineering manually is not necessary; this in turn can save me lots of time as the model can find the features that correlate and combine them for faster learning.

4.2 Classification

There are many different types of classification models which can be used to identify whether something is a botnet or not. One of these models is what is called a decision tree model. In a decision tree model, there is a classification starting at a root then testing attribute values at each node until it reaches a leaf node where a classification is given [9]. With something like this type of classification model. One of the main disadvantages of using binary trees for classification is that while they work very well with data that was used to create them, they are much more inaccurate when it comes to trying to be able to classify new samples of data. For this reason a binary tree classification model would not be very useful in a machine learning program as the main purpose of machine learning is to be able to classify data which is not the training data.

Another type of classification is called a Naïve Bayes. Naïve Bayes (also known as Multinomial Naïve Bayes in machine learning), this type of classification based on the probability a certain characteristic belongs to a certain class. An example of this would be looking at the words from an email to determine whether it is spam or not. In legitimate email, they might use certain words such as “Dear” and in spam emails words such as “Money” might come up more often. When training, the algorithm will build a “Histogram” of words that are contained within an E-mail, it will then build a table (of probabilities) of what words are likely to come up within that E-mail. When receiving incoming data, it will generate a separate score for both classifications depending on the probability tables, and which ever score is higher, the machine learning algorithm will classify it as that category. There also might be a problem with mis-classifying some data as a probability for something might be ‘0’. To counteract this, we use what is called a ‘Pseudo count’ which adds a number (typically one) to every bar in the histogram to avoid getting a 0 and in turn avoid any mis-classifications. One disadvantage of this is that it might treat the word “Dear Friend” the same way as it treats “Friend Dear”, however as with a botnet detection system, the order of the properties does not matter too much, but since we are working with lots of different numbers, it may lead to very large histograms and will be very computational inefficient.

Logistic Regression predicts whether something is true or false. It forms an ‘S’ curve which tells us how likely something is depending on a value, the higher the value, the more likely it is that characteristic. Logistic regression is usually used for classification by comparing a value on a

graph to see how likely it is to belong to a certain class (for example whether incoming traffic is a botnet or not). Similar to linear regression, we can also build models that take in multiple values to build more complex models to be able to make prediction much more accurately. An example of this might be calculating how obese a mouse is based on its weight, we might build a model that would say that a mouse has a certain percent chance of being obese based on a weight, and the machine learning aspect would be changing the threshold where a mouse would be considered obese based on to more accurately classify it.

Random Forest is another type of classification function that uses builds off decision trees. While normal decision trees are very static and only good at classifying data it has been trained off of, Random forest combines the simplicity of decision trees with flexibility. This results in much more accurate predictions. In a random forest tree, we need to create a bootstrapped data-set based off our original data-set. We then create a decision tree by picking what the nodes are randomly. We then repeat this multiple times until we have many trees. When testing new variables, we run the data we have down a tree and record the classification we get at the end of the tree, we then repeat this method for all the trees until we have gone through them all. After this we check the "tally" and then decide what the classification is. In a bootstrapped data-set we can have duplicates, this means that when we are testing the algorithm, we can use the original data-set which has been already labelled for us, for this reason testing the algorithm is quite easy as we do not need to get another data-set. For my botnet algorithm random forest would not be able to work properly, this is because decision trees are mostly binary decision and network traffic has a lot of data which is not binary, and while a threshold could have been implemented this could have taken a long time, such as the number of bytes a packet might be transferring or the date and time the traffic arrived and left with (which we can simplify to duration)

4.3 Loss Function

For the machine learning algorithm, there is a choice between many different types of loss functions. Loss functions help tell the algorithm how well it is performing on a data-set. Using this value, you can then make changes to the algorithm or experiment with different values and how you process the data you are given. It is a very useful value when building your algorithm and can help you in the process of creating a machine learning algorithm.

Another type of loss function is Mean absolute Error. This type of loss function looks at the difference between the actual value and the predicted value. With this type of loss function, it is not sensitive towards outliers and given several examples and the same input feature values, the optimal prediction will generally be the median value. One of the main disadvantages of this type of error loss is that the gradient magnitude is not dependent on the error size, meaning that even if

the error is quite small, this can lead to a convergence problem.

The 'Binary Cross-entropy' loss function is used in binary classification.This can also be used in multilabel problems where an example can belong to multiple classes at once, the model compares the example to each of the categories and then decides whether it belongs to that category or not. This can be used for music categorisation where certain songs can have multiple categories such as being part of the "Happy" category and "Energetic" Category. This type of loss function could be useful in my development of the algorithm as my main purpose is to identify whether incoming traffic is part of a botnet or not.

Categorical cross entropy is a type of loss function where an example can only belong to one category out of many different categories. This type of loss function is well suited for categorisation tasks since an example can belong to one specific category with a probability of 1 and to the other categories the probability can be 0. An example of this is the 'MNIST number recognition tutorial' where the number that is scanned can only be one of the numbers between 0 and 9.

After researching each of the loss functions, the loss function which will be used within the project will be 'Binary Cross-entropy' this is because we are testing for the category of whether a machine learning algorithm is a botnet or not, and this loss function seems to fit our needs the most.

4.4 Activation Function

The activation function is another part of an artificial neural network, as it allows the model to be able to recognise patterns within complex data. It is important when building your algorithm. An activation function is a function that defines an nodes output depending on its input.

Ridge functions are multi-iterative that act based on a linear combination of input variables, one example of this is a Linear activation function. In a ridge activation function, one advantage of this type of activation function is that Over-fitting can be prevented as having a data-set with lots of features can sometimes cause over fitting, this is because a model captures both real and random effects. Furthermore, rigid functions are also much simpler to understand and develop as an over complex model can sometimes be very hard to interpret. However, since you would need to build your model around a lower dimensional data-set, this can lead to a higher bias error.

Another type of activation function is a radial function. While a ridge function is a linear relationship between two variables, a ridge function has more of a curve between the axis. An example of this would be looking at dosages given to some test patient. Where a ridge function might try to highlight that more dosage leads to more effective results, a radial function would see more of a "bell curve" where it is most effective at medium dosage.

As the type of machine learning model that is being created is a classification model, deciding whether incoming traffic is a botnet or not, one very important activation function which is essential for this project. It is required for the final part of the model for the machine learning model to be able to properly classify incoming data.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

4.5 Optimiser Model

One part of a machine learning algorithm that can be experimented with is the Optimiser Model. In each of the experiments, the activation function for each of the layers was selected to be 'tanh'. The first model that had been experimented with is the Stochastic gradient decent (SDG). SGD is a build up from Gradient Descent Optimiser which also updates the weights iteratively, however in normal Gradient Descent, it updates these values only after seeing the whole data-set. This can lead to larger jumps meaning it may never reach the optimal value without being able to reach it [18]. SGD is different in the sense that it updates these values after each data-set allowing you to reach the optimal value much faster.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{4.2}$$

From this graph we see that the Train loss does fall fairly low, but the gradient decreases the more we train it. We also see some parts where the Train loss jumps up but the loss is slowly decreasing. Meanwhile the Test loss stays low throughout the entirety of the training of the algorithm. From this graph we can see that the algorithm isn't learning too quickly and given more time on the data-set the model does not seem to learn any better. There is also a risk of over fitting there the model learns how to identify botnets within the specific data-set rather than learning how to identify botnets itself. A reason for this slow learning rate is that as it is updating after sample, when it sees another sample which is drastically different the model would be confused by this and would not be able to correctly identify the next batch. One way to go around this is to update every N samples instead of every sample. The formula for SDG can be seen below.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{4.3}$$

Another type of optimisation model that had researched is the 'Adam' optimiser model. The adam optimiser is a combination of the 'adagrad' optimiser (or better known as the adaptive Loss modifier) and the RMSProp optimiser model [13]. This type of optimiser allows for adaptive learning

Figure 4.1: Schoastic Gradient Decent Optimiser Model Loss

Figure 4.2: Schoastic Gradient Decent Optimiser Model Loss

rates for separate parameters. Some of the advantages of adam optimiser model is that as it takes advantage of the adaptive loss modifier, the algorithm can learn at a much more accurate rate while taking advantage of the 'RMSProp' properties. This means that the learning rate is constantly changing depending on how well it is performing on the current batch of data.

We see better results in the optimisation model as we see the algorithm learning at a steady rate, we do see some spikes in the Train Loss. We can clearly see that this algorithm is learning much faster at learning than the 'SDG' Optimiser model. We also see some rises and falls in the train loss. Overall, the 'adam' optimiser model is a strong candidate in training an algorithm due to this strong learning.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{4.4}$$

4.6 Training Data

For the Training Data for the algorithm, a data-set which was already available and labelled was used for the training for the algorithm. Initially we was going to use a simulator to generate our own data-set and use this in the training algorithm. This had soon proven impractical as we needed data which had a mix of botnet data as well as legitimate traffic for the algorithm to learn. This on top of requiring the proper labelling of the data can lead to a very time-consuming process. Furthermore, proper labelling of the data would be required to accurately train the algorithm. This would require certain expertise within the information security field to be able to accurately label whether traffic is botnet or not. Furthermore, in order to be able to generate traffic data which has similar attributes to botnet data, expertise in the field is also required, and not having the proper experience would lead to a algorithm which has been trained on false data. In the end, pre-generated data was selected to be used for the data-set for the algorithm to be trained on. This comes with the advantage of being easy and cheap method, as well as having accurately labelled data for the algorithm to be trained one. However, we also have the disadvantage of this data-set being outdated as malicious attacks are always being updated, so the machine learning model may not be able to protect against the most recent attacks.

During the initial testing phases, a data-set that was publicly available was used for the algorithm [8]. With this data-set it needed to have some of the columns removed that the data had included, as the algorithm did not work with any string inputs, this means that columns such as Protocol or data such as IP needed to be removed from the data-set so the algorithm that could properly process the data frame of the input, as we want the algorithm to be trained on the packet data rather than where the packet is coming from. Another problem had risen that the contents of the Label column was classified as a string, one method of solving this is to alter the column, and find each of the rows which where classed as ”botnet”, and then replace that with a 1, and replace the other data which is not a botnet and replace that content to 0.

Chapter 5

Implementation of the Proposed Methods / Systems

5.1 Tools Used

For the language the algorithm will be developed on, it will be using Python 3. Python is a very powerful language when working with data making it a very popular choice for those working on machine learning algorithms. Furthermore, python has access to many libraries and frameworks which allow for data processing and data analysis much easier.

The work done on this project was also done using what is called a Jupyter Notebook. Jupyter Notebooks allow you to run codes in "Blocks" which makes experimenting with certain values much easier as you do not need to re run the entire program from the beginning again to fix a small error. The notebook also makes the organisation of code very intuitive as in one block you can have code which loads in the data, another block which alters the data-set, and another block to define the model architecture.

Dependencies were also relied upon for the success of this project, one of the dependencies which were used is Pandas. The Pandas library enabled us to extract data from a csv file to put into a variable. This library also enabled for manipulation of the extracted data-set in many ways which can prove very useful. One such way is the ability to filter out rows which contained 'N/A' values, which would otherwise interfere with the results of the machine learning algorithm.

NumPy is also an essential tool, as there are a few machine learning based libraries which depend on the usage of NumPy, this dependency is also very useful when when manipulating attributes of a table and converting them to a format which the machine learning algorithm can utilise and read

properly.

Sklearn (Shortened from scikit-learn) is a machine learning python library, it includes many different tools for machine learning, including classification, regression, clustering and dimensionality reduction. This library allowed the data-set that was loaded to be split into individual variables depending on whether the information is for training or testing the machine learning model.

Keras is another deep learning API, written for the python language. This library allows us to build machine learning models, quickly while being able to customise each layer to fit our needs, such as defining the number of nodes in each layer as well as the activation function of each one.

5.2 Dataset for Training

5.2.1 Data Acquisition

One method of acquiring data is the use of Honeypots, by using a real vulnerable machine or a virtual machine as bait, when infected it records data and give the network administrator a full view of the source and type of attack [16]. By using this constant source of information, we can then use this data to continually train a machine learning algorithm. One major advantage of this is by constantly updating our model by training it against real attacks, the model is always being exposed to the latest attack, ensuring the model is the best it can be. However, this can be an expensive and time consuming endeavor, as well as requiring much more intimate knowledge of internet security which is not within the scope of this paper.

Another method of data acquisition is by using freely available data-set online [8]. This comes with the main advantage of this being fast and cheap as the data has already been labelled by those who have much more knowledge and expertise within the field of botnets, so we can be reasonably confident that the data is as accurate as can be. However the main disadvantage being that the model will not be trained against the latest most dangerous threats.

After evaluation the strengths and drawbacks of both data acquisition methods, using a currently existing data-set as the training data is the best approach we can take for this machine learning project.

5.2.2 Data Cleaning

When working with real life network data, it is often the case that there will be lots of noise within said network traffic, for the machine learning algorithm to function at the highest capability possible, we must remove the noise. One way of cleaning up the data is by removing any data

which has a 'null' value attached to it, as it would otherwise severely affect the models performance if we were to directly use the data-set as the training data.

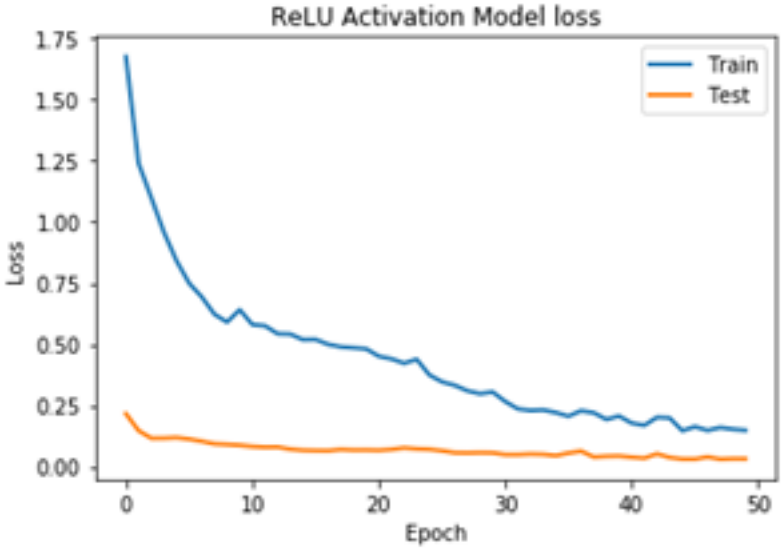
5.2.3 Feature Engineering

Furthermore, we also may want to remove certain data which has been captured in the data-set, one example of this is the Internet Protocol (IP) address in which the traffic is coming from, this is because we are not interested in the location the botnet is coming from, more-so the attributes of the traffic as this information is much more useful.

By removing features that are unnecessary, we can improve the efficiency of training as we are removing any unneeded input data as there are less variables for the machine learning algorithm to consider.

5.3 Activation Function

The next variable that was experimented on was the activation function of all the layers. In each case of the experiment, the optimiser model that was used was "adam". The loss function in each of the experimentation was also 'mean error squared' The first sets of tests were with the ridge activation functions. The two function that were chosen for this experiment were the "linear" and the "ReLU" activation functions. These results can be seen below



In the case of linear activation functions we can see that initially the model is learning fairly quickly

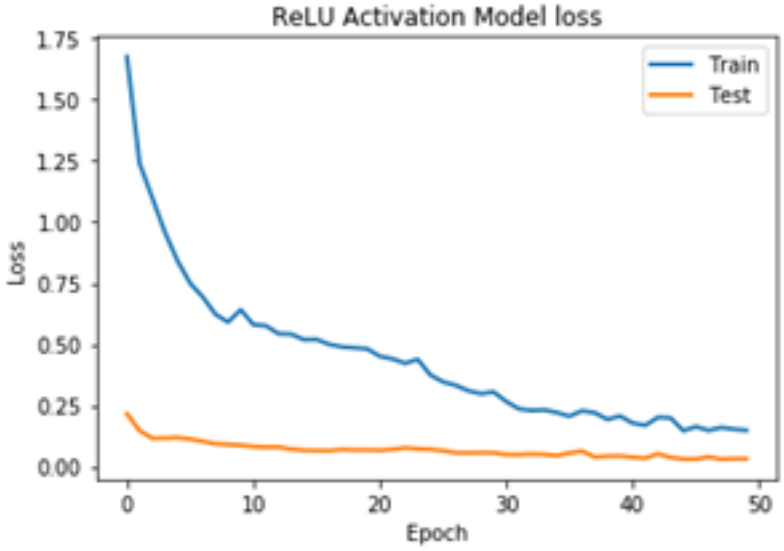


Figure 5.1: Relu Activation Function Model Loss

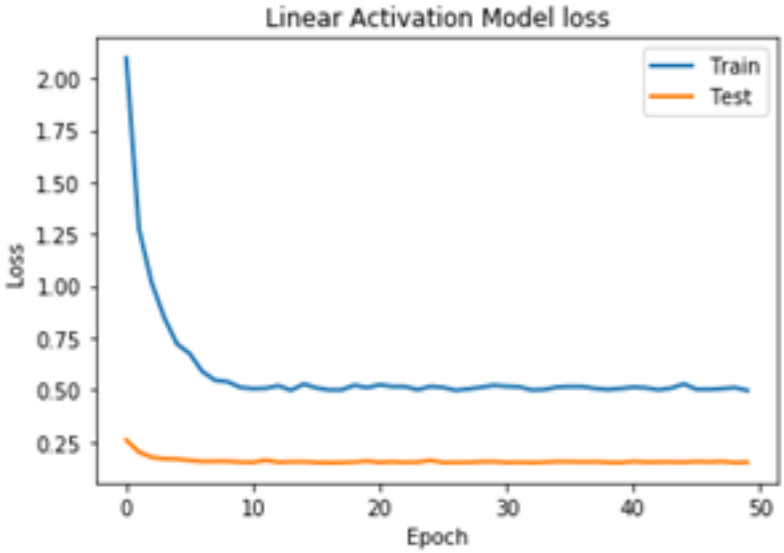


Figure 5.2: Linear Activation Model Loss

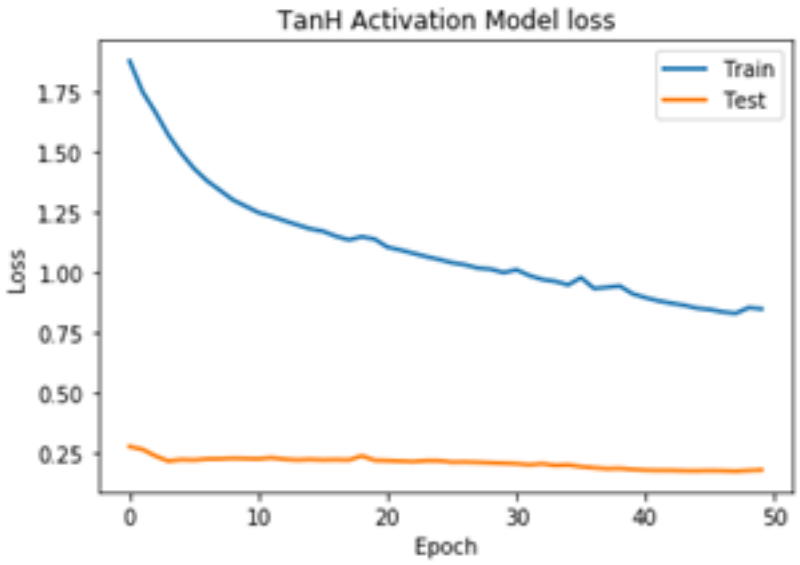


Figure 5.3: Linear Activation Model Loss

however when it reaches a certain value, the loss hovers around a certain value, this can be because the model is making its "jumps" which are too large for the model to be able to reach its optimal value. On the other hand, we can see that the ReLU model is learning at a much better time reaching the optimal value as it is adjusting the size of the "jumps" the model makes.

The next set of tests were with the radial functions. In this set of tests, the 'tanh' and the 'rbf2' activation functions will be experimented. While the 'tanh' function has already been experimented with, it needs to be retested with similar variables (e.g the number of epochs)

The tanh activation function has a hard time learning how to identify botnets within a network, but we still see a downwards trend so we should expect to see it performing better given the extra time to train. Compared to the other activation models, this one does not decrease its loss as quickly.

Chapter 6

Testing and Integration

In this chapter, we will be testing the data-set and ensuring that it is fit for purpose for the machine learning algorithm can train off the data without any issues.

6.1 Data-set Array Size

For the machine learning array to function properly, the array shape should be the right size to ensure the proper training of the model, the input array should high dimensional, to ensure that all the features are inputted properly, where as the output array size should only be one-dimensional, this is because the outcome of this project is a classification task, checking whether incoming traffic is a botnet or not. Both input and output arrays have been outputted, and the input arrays are of high-dimension arrays and the output arrays are of one-dimensional arrays. This shows that the data-set are ready for the machine learning algorithm.

6.2 Data Pre-processing

Furthermore, we should ensure the data-set is split into train and test variables. We need to do this to ensure that the model has data to train on, as well as unseen data to test how effective the model has learned from the data-set, as any data the model may receive afterwards will be unseen. Furthermore, we must ensure that there are no null/NaN values within the database that will affect, we can test to see if there are any null values that exist within the array and output ”true” if any of the values contain a null or ’NaN’. None of the data after processing contains ’NaN’ or ’null’ values, so we know it is fit for purpose.

Figure 6.1: Output of arrays used for testing

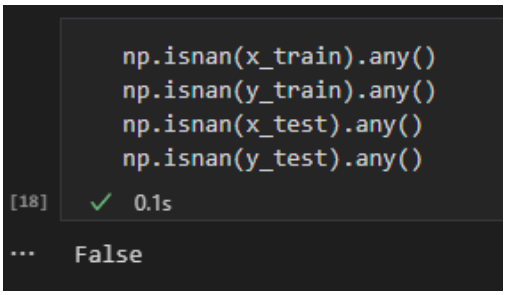


Figure 6.2: Test to see if there are null/NaN values in our data-set

6.3 Data Sampling and Labelling

We can also test to see if the data-set we are inputting into the system is properly labelled as well as containing all the input columns that we wish to use for training. In this case, code is ran to update all the labels containing the word ”botnet” to 1, while all of the other labels will be assigned to 0. Then we can take a sample of the data-set to ensure that the labelling is done correctly and all the input data which we wish to use for training is in the data-set.

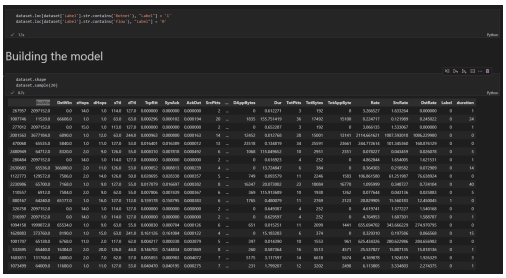


Figure 6.3: Sample of Data-set, along with Label

Chapter 7

Product Evaluation

In this chapter, we describe the plan and design of the product evaluation and their results. The purpose of the evaluation is to evaluate to what extent the product has achieved the aim defined in the introduction section. We must explain what we have evaluated and measured, and WHY. Also, you will explain the results, and their interpretation. Lastly, we discuss the results. Specifically, you must objectively conclude whether you succeeded in achieving the initial aim, you failed, or the evaluation was not sufficient to prove it. Although discussion is usually given in an independent section out of the result section in natural science areas, it is common to put them in the same section in computer science areas. Still, we need to make sure to avoid confusion between the results and the discussion. In a data science project, the evaluation may include objective metrics, such as the mean squared error for regression, the accuracy score and the F1 score in classification, the adjusted Rand score in clustering. In a software engineering project, the evaluation may include user survey.

7.1 Accuracy

Accuracy is one method of evaluation how successful a machine learning model, it is calculated by adding up the 'True Positive' and 'True Negative', and then dividing that number by the total number of predictions. Accuracy is a good baseline in evaluating a machine learning model, however it should not be taken alone when evaluating how successful a model is.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

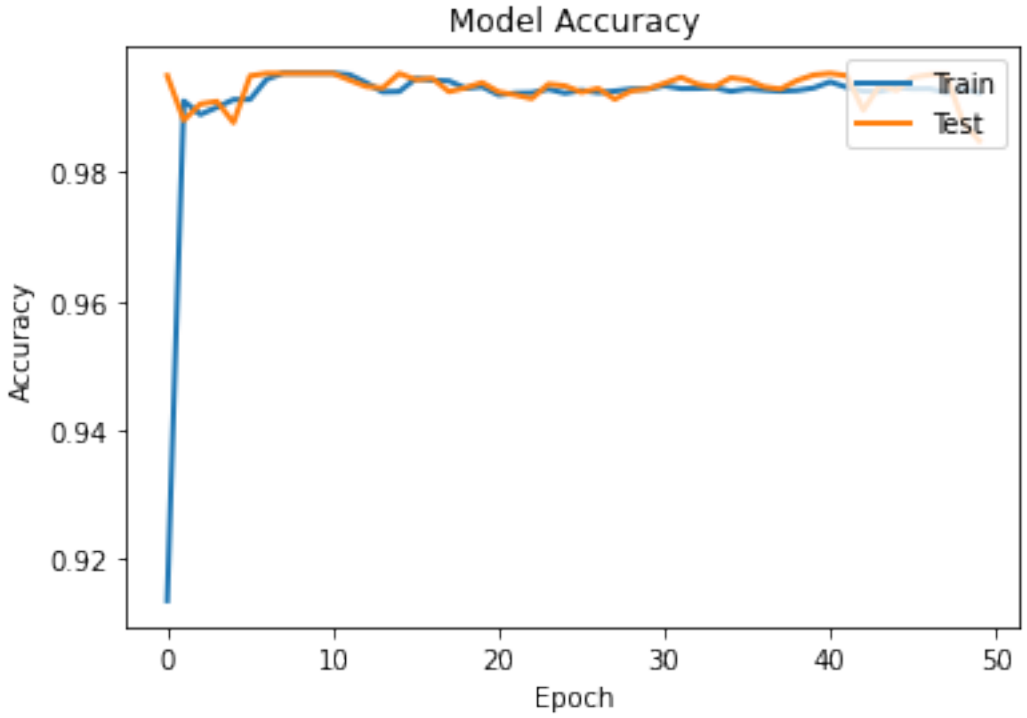


Figure 7.1: Botnet category classification accuracy on Data-set[8]

After training the machine learning model, we ended up with an accuracy of 99.5%. This accuracy has exceeded the target accuracy of 90% established earlier in the paper in its aims and objectives.

7.2 Mean Squared Error

Mean Squared Error is another metric we can evaluate how successful a machine learning algorithm model is. This tells you how close a regression line is to a set of points, the lower this value is, the better the model is at predicting the correct value. This is a very useful metric in analysing the effectiveness of a machine learning model as we can see any errors amplified as the values are being squared.

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{7.2}$$

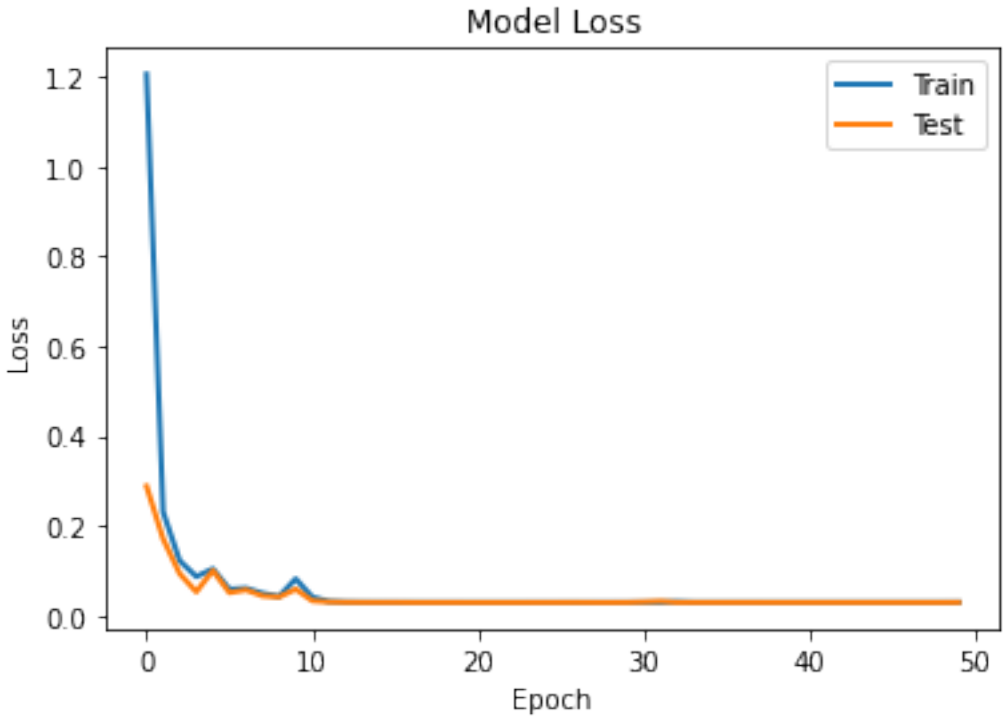


Figure 7.2: Machine learning Model Loss

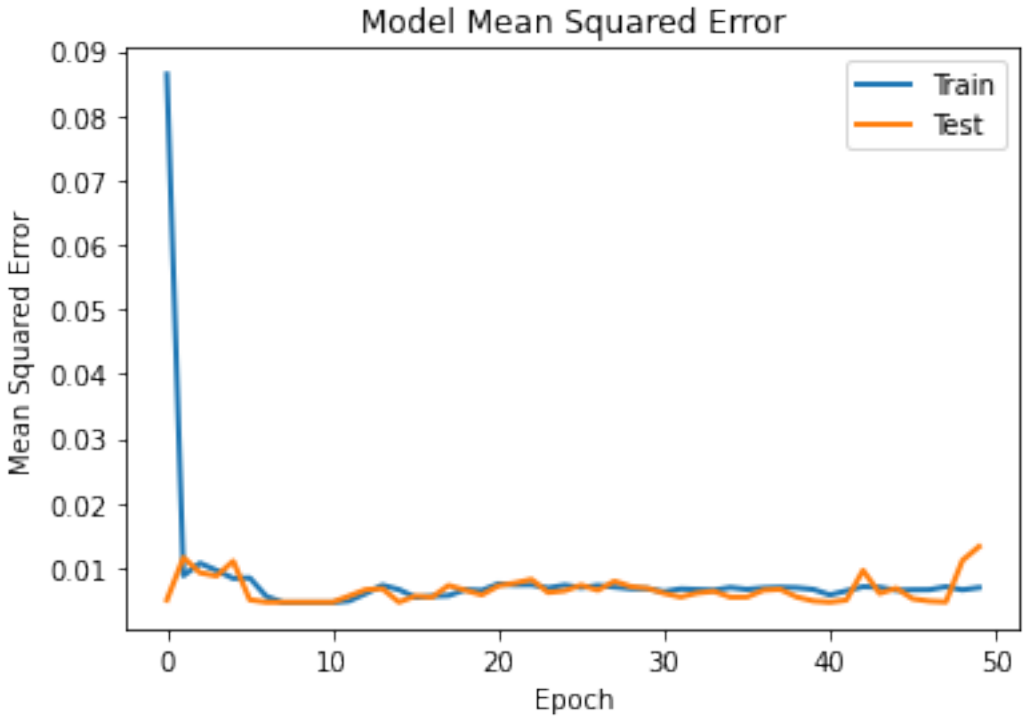


Figure 7.3: Mean Squared Error of model on Data-set[8]

Chapter 8

Closing chapter

The closing chapters commonly include a summary and a conclusion together with any recommendations. In summarising, highlight the important stages and outcomes of the project. The conclusions would normally consider and comment critically upon the results of the project; this includes both the process and the product. This should include a consideration of the extent to which the aims of the project have been achieved. Finally, recommend ways in which the work could be applied or extended. All of the discussion here must be objective, based on the test and evaluation results. Readers do not need our personal or subjective opinions.

8.1 Evaluation

After evaluating the product, we can conclude that this project was an overall success. One important stage we can discuss is the importance of making sure the data-set we are using to train the model is suitable for the model to be able to train effectively. Not taking extra-care in preparing the data-set to be used for the machine learning model can be detrimental to a project so sufficient time should be set apart in order to be able to ensure the success of the machine learning project.

8.2 Problems faced Lessons Learned

During the construction of this project, one of the biggest issued faced was to do with the data-set and the pre-processing of the data-set. In the initial stages of the project, many different data-sets were used to train the machine learning algorithm. Many of these resulted in the model not being able to learn properly and having low accuracy. One of the major reasons of this was due to not processing the data-set properly as the output of the data-set was high-dimensional when it was

required to be one-dimesional. This experience shows us the importance of properly processing data, as if this step is not done properly, it will result in poor results.

8.3 Improvements

This project can be improved further by training it on more data-sets. This is because the model that has been trained has only trained on a specific data-set that has been acquired from the internet, this would result in the model being limited in identifying botnets as it will only be able to identify attacks which have been captured within the data-set and not any that are new attacks or some which have not been covered within the data-set. Furthermore, more testing of the machine learning model could have been done as we are only using one data-set to both train as well as test the model, this may lead to some over fitting where the model is learning the data-set rather identifying botnet packets within network traffic.

8.4 Future Work

One method in which this project can be taken further is by classifying the type of botnet. As this model only checks whether incoming traffic is a botnet or not, it can be improved further to identify the type of attack associated with the botnet. One example of this is identifying if a botnet intends to distribute a phishing email within a system, once a phishing botnet has been detected, counter measures such as alerting all staff to be extra weary of any emails they may receive.

Bibliography

[1] Divya Arora, Mehak Garg, and Megha Gupta. Diving deep in deep convolutional neural network. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 749–751, 2020.

[2] Ahmet Tuğrul Bayrak, Asmin Alev Aktaş, Orkun Susuz, and Okan Tunalı. Churn prediction with sequential data using long short term memory. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–4, 2020.

[3] Deepak R Bharadwaj, Anamika Bhattacharya, and Manivannan Chakkaravarthy. Cloud threat defense – a threat protection and security compliance solution. In *2018 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 95–99, 2018.

[4] S. Chen, Y. Chen, and W. Tzeng. Effective botnet detection through neural networks on convolutional features. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 372–378, 2018.

[5] M. Debashi and P. Vickers. Sonification of network traffic for detecting and learning about botnet behavior. *IEEE Access*, 6:33826–33839, 2018.

[6] X. Dong, J. Hu, and Y. Cui. Overview of botnet detection based on machine learning. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 476–479, 2018.

[7] Joanita D'souza, Maninder Jeet Kaur, Hassan Anwar Mohamad, and Piyush Maheshwari. Transmission control protocol (tcp) delay analysis in real time network. In *2020 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6, 2020.

[8] Sebastian Garcia. Malware capture facility project.

[9] Sachin S. Gavankar and Sudhirkumar D. Sawarkar. Eager decision tree. In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pages 837–840, 2017.

[10] G. Goth. Functionality meets terminology to address network security vulnerabilities. *IEEE Distributed Systems Online*, 7(6):4–4, 2006.

[11] Shifeng Huang. Influence of different convolutional neural network settings on the performance of mnist handwritten digits recognition. In *2020 International Conference on Artificial Intelligence and Education (ICAIE)*, pages 1–6, 2020.

[12] Yimu Ji, Lu Yao, Shangdong Liu, Haichang Yao, Qing Ye, and Ruchuan Wang. The study on the botnet and its prevention policies in the internet of things. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 837–842, 2018.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[14] Caiming Liu, Yan Zhang, Zhonghua Li, Jiandong Zhang, Hongying Qin, and Jinquan Zeng. Dynamic defense architecture for the security of the internet of things. In *2015 11th International Conference on Computational Intelligence and Security (CIS)*, pages 390–393, 2015.

[15] Vasileios Mavroeidis and Siri Bromander. Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In *2017 European Intelligence and Security Informatics Conference (EISIC)*, pages 91–98, 2017.

[16] Vasileios A. Memos and Kostas E. Psannis. Ai-powered honeypots for enhanced iot botnet detection. In *2020 3rd World Symposium on Communication Engineering (WSCE)*, pages 64–68, 2020.

[17] Beny Nugraha, Anshitha Nambiar, and Thomas Bauschert. Performance evaluation of botnet detection using deep learning techniques. In *2020 11th International Conference on Network of the Future (NoF)*, pages 141–149, 2020.

[18] John Pomerat, Aviv Segev, and Rituparna Datta. On neural network activation functions and optimizers in relation to polynomial regression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6183–6185, 2019.

[19] Hanyong Shao. Delay-dependent stability for recurrent neural networks with time-varying delays. *IEEE Transactions on Neural Networks*, 19(9):1647–1651, 2008.

[20] Peng Shengguang. Overview of meta-reinforcement learning research. In *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pages 54–57, 2020.

[21] Sachchidanand Singh and Nirmala Singh. Internet of things (iot): Security challenges, business opportunities reference architecture for e-commerce. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1577–1581, 2015.

[22] Junyoung Son, Jonggyun Choi, and Hyunsoo Yoon. New complementary points of cyber security schemes for critical digital assets at nuclear power plants. *IEEE Access*, 7:78379–78390, 2019.

[23] Chanin Tansettanakorn, Supachai Thongprasit, Siritam Thamkongka, and Vasaka Visoottiviseth. Abis: A prototype of android botnet identification system. In *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pages 1–5, 2016.

[24] R. Vishwakarma and A. K. Jain. A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1019–1024, 2019.

[25] Gernot Vormayr, Tanja Zseby, and Joachim Fabini. Botnet communication patterns. *IEEE Communications Surveys Tutorials*, 19(4):2768–2796, 2017.

[26] Yubin Wang, Chao Li, and Nan Cheng. Internet security protection in personal sensitive information. In *2014 Tenth International Conference on Computational Intelligence and Security*, pages 628–632, 2014.

[27] Ruyue Xin, Jiang Zhang, and Yitong Shao. Complex network classification with convolutional neural network. *Tsinghua Science and Technology*, 25(4):447–457, 2020.

[28] Nilay Yildirim and Asaf Varol. A research on security vulnerabilities in online and mobile banking systems. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5, 2019.

[29] Zonghua Zhang, Ruo Ando, and Youki Kadobayashi. Hardening botnet by a rational botmaster. In Moti Yung, Peng Liu, and Dongdai Lin, editors, *Information Security and Cryptology*, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.