|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Testing Detection Accuracy of IOT Threats With Deep Learning Techniques  Project Update V1.0 |
| |  |  |  | | --- | --- | --- | | Badruddoja, Syed | Hicks, Justin | Santamaria, Keith | 4/11/20 | CSCE5933:Deep Learning | |

Table of Contents

[Introduction 2](#_Toc37514232)

[Project Organization & Team 2](#_Toc37514233)

[Related projects 2](#_Toc37514234)

[Figure 1: An illustration of TOR communication between Alice and a destination server. 2](#_Toc37514235)

[Problem Definition 3](#_Toc37514236)

[Figure 2: The Testbed configuration for the BoT-IoT Dataset from UNSW Canberra 3](#_Toc37514237)

[Project Design and Milestones: 4](#_Toc37514238)

[Design Methods 4](#_Toc37514239)

[Figure 3: General Design of prediction for IOT BOT Dataset 4](#_Toc37514240)

[Training Model 4](#_Toc37514241)

[Regularization 4](#_Toc37514242)

[Activation Function: 4](#_Toc37514243)

[Optimized Model: 5](#_Toc37514244)

[Tools & Libraries 5](#_Toc37514245)

[Dataset 5](#_Toc37514246)

[Table 1: Feature details 6](#_Toc37514247)

[Project Plan: 7](#_Toc37514248)

[Implementation 8](#_Toc37514249)

[Label Encoding 8](#_Toc37514250)

[Table 2: Label Encoding References 8](#_Toc37514251)

[Preprocessing 8](#_Toc37514252)

[Base Neural Networks 8](#_Toc37514253)

[Optimizers 8](#_Toc37514254)

[Regularization 8](#_Toc37514255)

[Results: 9](#_Toc37514256)

[Graph 1: Regular CNN Training and testing accuracy 9](#_Toc37514257)

[Graph 2: Regularized CNN training & testing Graph 9](#_Toc37514258)

[Graph 3: ADAM Optimized CNN training & testing Accuracy 10](#_Toc37514259)

[Table 3: Average Accuracy Comparison for CNN Model 10](#_Toc37514260)

[Conclusion 10](#_Toc37514261)

[References 10](#_Toc37514262)

# Introduction

Distributed denial of services attacks (DDoS attacks) is one of the most common network attacks that occurs. An average of 28,700 DDoS attacks occur every day, with an average cost of $40,000- $50,000 per hour. With how serious and frequent of a financial threat these are, it is extremely important for security models to detect these botnets before they can crash a network.

BoT-IoT dataset was created by designing a realistic network environment in the Cyber Range Lab of The center of UNSW Canberra Cyber, as shown in Figure 1

# Project Organization & Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Member** | **Email** | **Responsibility** | **Meeting Schedule** | **Team Adviser** |
| 1 | Syed Badruddoja | syedbadruddoja@  my.unt.edu | Perform CNN and Documentation | Every thursday after class  online via discord | Dr. Mark Albert |
| 2 | Keith Santamaria | [keithsantamaria@ my.unt.edu](mailto:keithsantamaria@my.unt.edu) | Perform RNN & Documentation |
| 3 | Justin Hicks | [justinhicks@ my.unt.edu](mailto:justinhicks@my.unt.edu) | Perform CNN and Documentation |

Project Repository: <https://github.com/JJHicks/IoT-Botnet-Attack-Detection>

# Related projects

[8] discusses about ways to detect Tor traffic using deep learning methods. Tor traffic is kind of a encrypted tunnel between the client and a vpn server which is hidden from internet and it is hard to trace them without any specialized software.

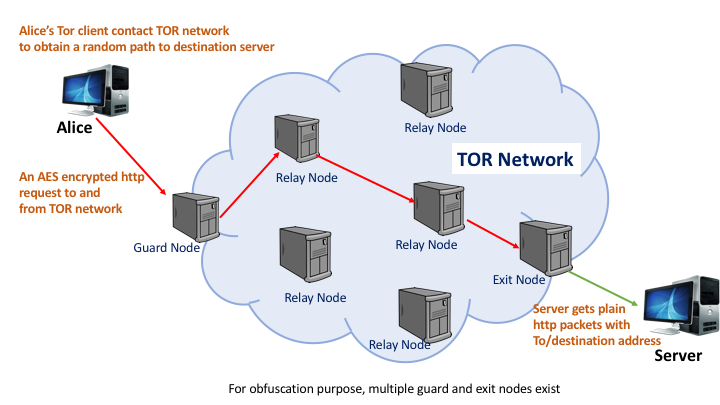


Figure 1: An illustration of TOR communication between Alice and a destination server.

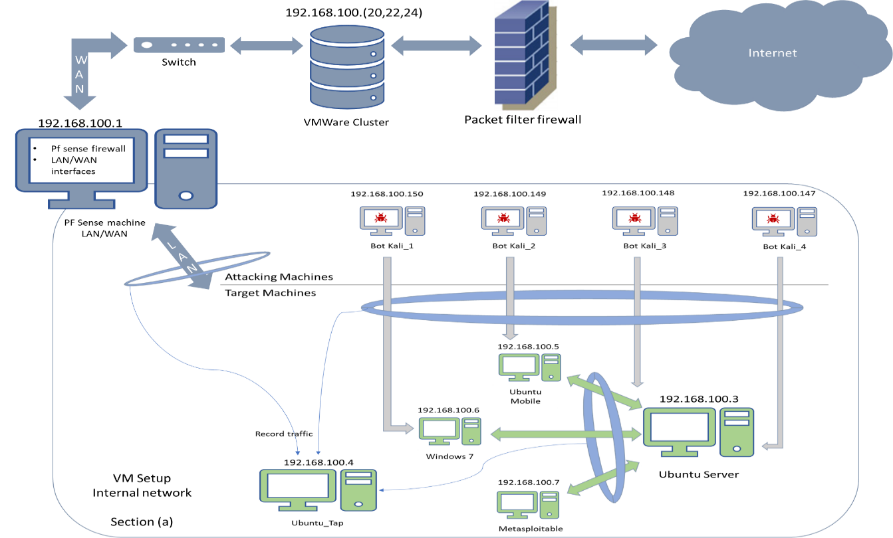
*“The communication starts with Alice requesting a path to the server. TOR network gives the path which is AES encrypted. The randomization of the path happens inside the TOR network. The encrypted path of the packet is shown in red. Upon reaching the exit node, which is the periphery node of the TOR network, the plain packet is transferred to the server.”*

[4] discussed about A Deep Learning Approach for Botnet Traffic Detection. While botnets have been extensively studied, bot malware is constantly advancing and seeking to exploit new attack vectors and circumvent existing measures. Existing intrusion detection systems are unlikely to be effective countering advanced techniques deployed in recent botnets. This chapter proposes a deep learning-based botnet traffic analyser called Botnet Traffic Shark (BoTShark). BoTShark uses only network transactions and is independent of deep packet inspection technique; thus, avoiding inherent limitations such as the inability to deal with encrypted payloads. This also allows us to identify correlations between original features and extract new features in every layer of an Autoencoder or a Convolutional Neural Networks (CNNs) in a cascading manner. Moreover, we utilise a Softmax classifier as the predictor to detect malicious traffics efficiently. © Springer International Publishing AG, part of Springer Nature 2018

[6] is about Recurrent neural networks for Cyber security. Recurrent neural network (RNN) is an effective neural network in solving very complex supervised and unsupervised tasks. There has been a significant improvement in RNN field such as natural language processing, speech processing, computer vision and other multiple domains. This paper deals with RNN application on different use cases like Incident Detection, Fraud Detection, and Android Malware Classification. The best performing neural network architecture is chosen by conducting different chain of experiments for different network parameters and structures. The network is run up to 1000 epochs with learning rate set in the range of 0.01 to 0.5.Obviously, RNN performed very well when compared to classical machine learning algorithms. This is mainly possible because RNNs implicitly extracts the underlying features and also identifies the characteristics of the data. This helps to achieve better accuracy.

# Problem Definition

Our aim is to create a deep learning model that can accurately distinguish legitimate network traffic from an IoT botnet attack. The dataset was created by designing a realistic network environment that generates both normal and botnet traffic. The dataset consists of features made from fields in packet information, like what you would find using Wireshark, along with labels for the attack categories and subcategories. The benefit of generating this data in a controlled environment is that it addresses the issues of incomplete network information and inaccurate labeling, in addition to giving better data for more complex and diverse attacks.

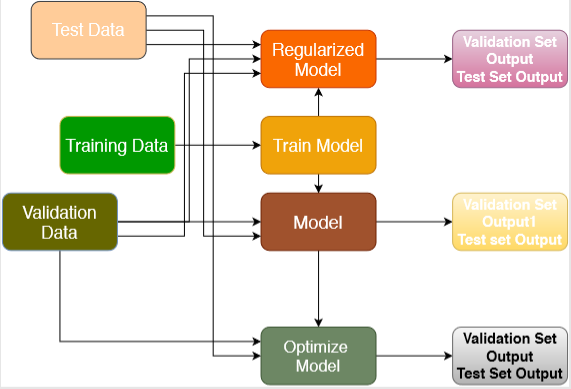


### Figure 2: The Testbed configuration for the BoT-IoT Dataset from UNSW Canberra

# Project Design and Milestones:

Project design is made simple to test the neural network deep learning methods to test the performance and check feasibility of suggesting deep learning method as suitable for cyber security threat predictions.

## Design Methods



### Figure 3: General Design of prediction for IOT BOT Dataset

### Training Model

The dataset will go through the training phases of fitting through a model of CNN. Initially we will be using less number of neurons at 2 layers to check the accuracy of the model, later on we will be checking for more results

### Regularization

Noise robustness will be tested for accuracy differences and a comprehensive study is planned for this.

### Activation Function:

Rectified linear unit activation function will be used for making the outputs positives and have a comprehensive understanding of the data. We will explore more on this in coming days

### Optimized Model:

Adam Optimization is an optimization technique which is a combination of RMSprop, Stochastic gradient descent and momentum. We have planned to implement this and make adjustments to see how the accuracy differs.

Figure 2 describes the flow chart that will be used for this project as a basic model. We will be using several deep learning models to study how cyber security attacks can be predicted out of the test data. .

## Tools & Libraries

As per current planning, we would like to perform our deep learning prediction with tensorflow and keras packages in a local machine that has good ram to perform the operation on a reasinable time. If we get time, we would like to perform the same on google cloud platform to test the prediction with respect to time and accuracy.

## Dataset

The IoT Botnet dataset comes in a series of 75 csv files totaling ~15GB. We plan to use this, Tensorflow (Python), and Git for source control to get started in setting up our first model. We will update this section as we progress into the project.For hardware accelerated model training, we have a Nvidia GTX 1080 and 1060 available

The features we have in our dataset are enumerated below.

|  |  |
| --- | --- |
| Feature | Description |
| pkSeqID | Row Identifier |
| Stime | Record start time |
| Flgs | Flow state flags seen in transactions |
| flgs\_number | Numerical representation of feature flags |
| Proto | Textual representation of transaction protocols present in network flow |
| proto\_number | Numerical representation of feature proto |
| Saddr | Source IP address |
| Sport | Source port number |
| Daddr | Destination IP address |
| Dport | Destination port number |
| Pkts | Total count of packets in transaction |
| Bytes | Totan number of bytes in transaction |
| State | Transaction state |
| state\_number | Numerical representation of feature state |
| Ltime | Record last time |
| Seq | Argus sequence number |
| Dur | Record total duration |
| Mean | Average duration of aggregated records |
| Stddev | Standard deviation of aggregated records |
| Sum | Total duration of aggregated records |
| Min | Minimum duration of aggregated records |
| Max | Maximum duration of aggregated records |
| Spkts | Source-to-destination packet count |
| Dpkts | Destination-to-source packet count |
| Sbytes | Source-to-destination byte count |
| Dbytes | Destination-to-source byte count |
| Rate | Total packets per second in transaction |
| Srate | Source-to-destination packets per second |
| Drate | Destination-to-source packets per second |
| TnBPSrcIP | Total Number of bytes per source IP |
| TnBPDstIP | Total Number of bytes per Destination IP. |
| TnP\_PSrcIP | Total Number of packets per source IP. |
| TnP\_PDstIP | Total Number of packets per Destination IP. |
| TnP\_PerProto | Total Number of packets per protocol. |
| TnP\_Per\_Dport | Total Number of packets per dport |
| AR\_P\_Proto\_P\_SrcIP | Average rate per protocol per Source IP. (calculated by pkts/dur) |
| AR\_P\_Proto\_P\_DstIP | Average rate per protocol per Destination IP. |
| N\_IN\_Conn\_P\_SrcIP | Number of inbound connections per source IP. |
| N\_IN\_Conn\_P\_DstIP | Number of inbound connections per destination IP. |
| AR\_P\_Proto\_P\_Sport | Average rate per protocol per sport |
| AR\_P\_Proto\_P\_Dport | Average rate per protocol per dport |
| Pkts\_P\_State\_P\_Protocol\_P\_DestIP | Number of packets grouped by state of flows and protocols per destination IP. |
| Pkts\_P\_State\_P\_Protocol\_P\_SrcIP | Number of packets grouped by state of flows and protocols per source IP. |
| Attack | Class label: 0 for Normal traffic, 1 for Attack Traffic |
| Category | Traffic category |
| Subcategory | Traffic subcategory |

### Table 1: Feature details

## Project Plan:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Task | Description | Work Type | Deadline | Status |
| 1 | Get Dataset and Filter dataset | Study relevant dataset if required to add or delete | Coding and Documentation | 4th April | Completed |
| 2 | Normalize dataset | using normalization technique if required | Coding and Documentation | 4th April | Completed |
| 3 | label encoding | Classify objects as numbers, we can also use one hot encoding | Coding and Documentation | 4th April | Completed |
| 4 | Perform Basic ANN Learning | Study time, accuracy and relevant metrics if possible | Coding and Documentation | 5th April | Ongoing |
| 5 | Perform CNN Prediction | Not sure if this one is possible | Coding and Documentation | 6th april | Completed |
| 6 | Perform RNN prediction | Study time, accuracy and relevant metrics if possible | Coding and Documentation | 7th April | Ongoing |
| 7 | Perform Regularization | Study time, accuracy and relevant metrics if possible | Coding and Documentation | 8th April | Partially Done |
| 8 | Perform Optimization | Study time, accuracy and relevant metrics if possible | Coding and Documentation | 8th April | Partially Done |
| 9 | Study loss and Accuracy Behaviour | Compare the results | Documentation | 9th April | Ongoing |
| 10 | Additional reports | Addition to what we have done, like autoencoders | Coding and Documentation | 9th April | Ongoing |
| 11 | Prepare Project update | All deails of project should be ready | Documentation | 10th April | Completed |
| 12 | Prepare Final Project Report | Includes everything with metrics and results | Documentation | 18th April | Ongoing |
| 13 | Prepare Final Project Presentation | Prepare a complete documentation similar to a conference paper if possible | Documentation | 18th April | Ongoing |

# Implementation

## Label Encoding

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Service State/ Encoding | Service/Encoding | Attack/Encoding |
| 1 | CON/1 | dhcp / 1 | generic / 0 |
| 2 | FIN/2 | dns / 2 | normal / 1 |
| 3 | INT/3 | ftp / 3 | Analysis / 2 |
| 4 | REQ/4 | ftpprotocoldata / 4 | Backdoor / 3 |
| 5 | RST/5 | http / 5 | Dos / 4 |
| 6 | ACC/6 | irc / 6 | exploits / 5 |
| 7 | CLO-7 | pop3 / 7 | Fuzzers / 6 |
| 8 | NA | protocol / 8 | Reconnaissance / 7 |
| 9 | NA | radius / 9 | shellcode / 8 |
| 10 | NA | snmp / 10 | worms / 9 |
| 11 | NA | smtp / 11 | NA |
| 12 | NA | ssh / 12 | NA |
| 13 | NA | ssl / 13 | NA |

### Table 2: Label Encoding References

## Preprocessing

[Under Construction]

## Base Neural Networks

[Under Construction]

## Optimizers

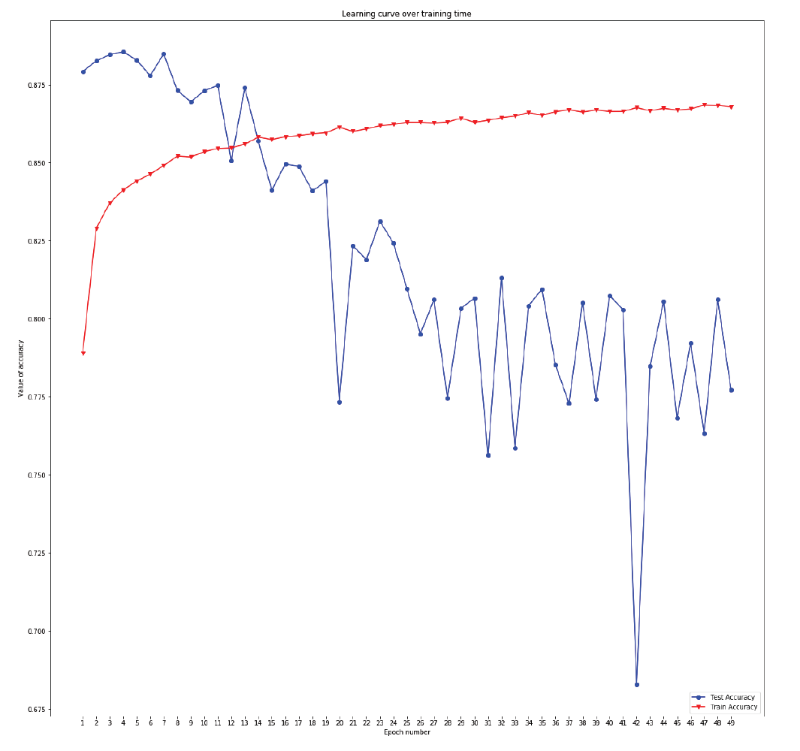
[Under Construction]

## Regularization

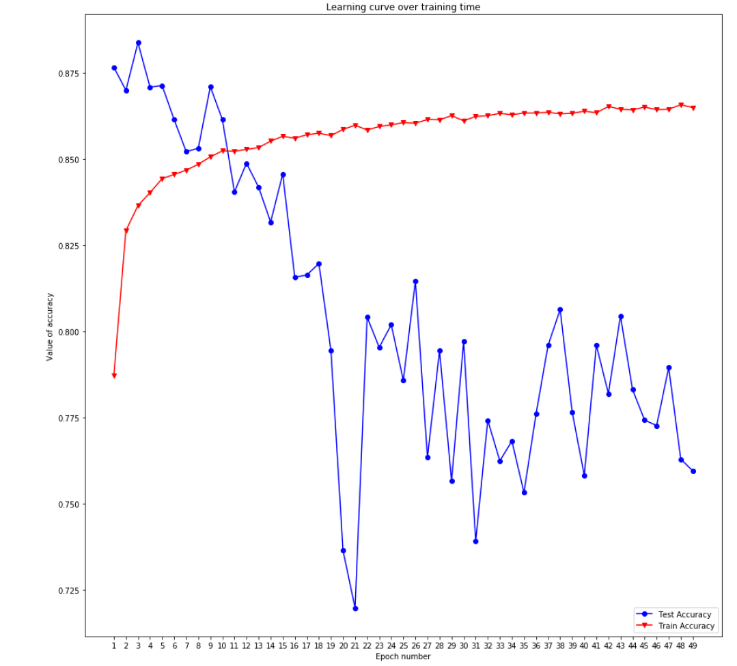
[Under Construction]

# Results:

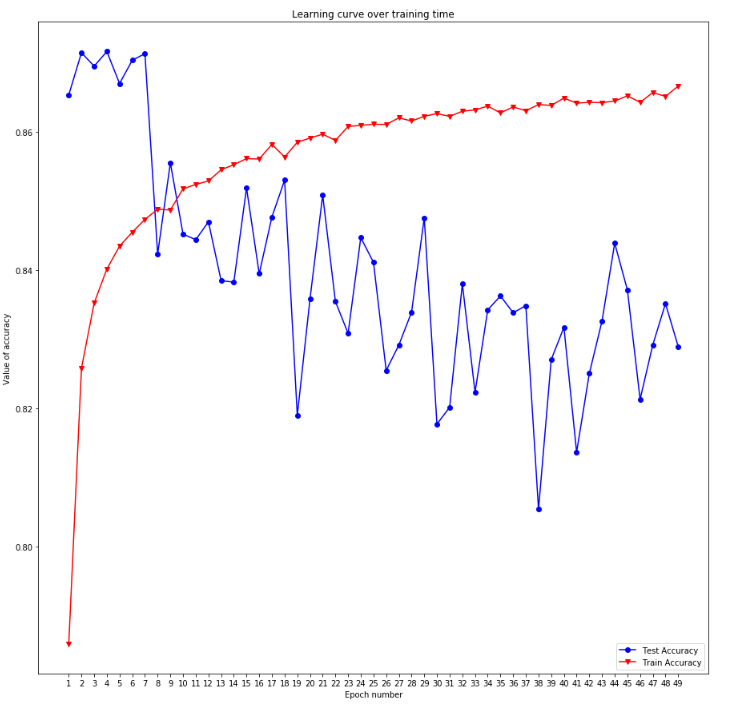
[Under Construction]



### Graph 1: Regular CNN Training and testing accuracy



### Graph 2: Regularized CNN training & testing Graph



### Graph 3: ADAM Optimized CNN training & testing Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| No | CNN Accuracy(Average) | Regularized CNN (Average) | Optimized CNN(Average) |
| Accuracy |  |  |  |
| Loss |  |  |  |

### Table 3: Average Accuracy Comparison for CNN Model

# Conclusion

[Under Construction]

# References

1. <https://arxiv.org/abs/1811.00701> - The paper regarding the creation and usefulness of our dataset
2. <https://hostingtribunal.com/blog/ddos-statistics/#gref> - DDoS attack statisticsA
3. <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/index.php> - a dataset that the webpage of your dataset recommended comparision with.
4. BoTShark: A Deep Learning Approach for Botnet Traffic Detection, Ref : <https://www.researchgate.net/publication/324700685_BoTShark_A_Deep_Learning_Approach_for_Botnet_Traffic_Detection>
5. Forensics and Deep Learning Mechanisms for Botnets in Internet of Things: A Survey of Challenges and Solutions, Ref:“<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8713986>”
6. RNNSecureNet: Recurrent neural networks for Cyber security use-cases,Ref: ”<https://arxiv.org/abs/1901.04281>”
7. Using the Power of Deep Learning for Cyber Security ,Ref:”https://www.analyticsvidhya.com/blog/2018/07/using-power-deep-learning-cyber-security/”
8. Using the Power of Deep Learning for Cyber Security (Part 2)<https://www.analyticsvidhya.com/blog/2019/05/using-power-deep-learning-cyber-security-2/>
9. Adam Optimization : <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>