Deep learning for Domain Generation Algorithm

PHASE 2 REPORT

Submitted by

Dinesh Kumar C
(RCAS2021MDB070)

in partial fulfillment for the award of the degree of

MASTER OF SCIENCE
IN
DATA SCIENCE AND BUSINESS ANALYTICS

DEPARTMENT OF COMPUTER SCIENCE

RATHINAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

COIMBATORE - 641021 (INDIA)

MAY - 2023

RATHINAM COLLEGE OF ARTS AND SCIENCE
(AUTONOMOUS)
COIMBATORE - 641021

BONAFIDE CERTIFICATE

This is to certify that the thesis entitled **Deep learning for Domain Generation Algorithm** submitted by **Dinesh Kumar C (RCAS2021MDB070)**, for the award of the Degree of Master of Computer Science in **"DATA SCIENCE AND BUSINESS ANALYTICS"** is a bonafide record of the work carried out by him under my guidance and supervision at Rathinam College of Arts and Science, Coimbatore

Er.M.Saravanakumar MCA.,ME(CSE)., — **Dr.SivaPrakash M.Tech.,(Ph.D)**

Supervisor — Mentor

Submitted for the university examination held on

INTERNAL EXAMINER — **EXTERNAL EXAMINER**

RATHINAM COLLEGE OF ARTS AND SCIENCE
(AUTONOMOUS)
COIMBATORE - 641021

DECLARATION

I'm **Dinesh Kumar C (RCAS2021MDB070)**, hereby declare that this Phase 2 entitled **" Deep learning for Domain Generation Algorithm",** is the record of the original work done by me under the guidance of **Er.M.Saravanakumar MCA., ME(CSE)**,Senior Faculty-IT Rathinam college of arts and science, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree/diploma/ associateship/fellowship/or a similar award to any candidate in any University.

Dinesh Kumar C

Signature of the Student

Place: Coimbatore

Date: 01.12.2022

COUNTERSIGNED

Er.M.Saravanakumar MCA.,ME(CSE)., MISTE
Supervisor

Contents

Acknowledgement

On successful completion for project look back to thank who made in possible. First and foremost, thank **"THE ALMIGHTY"** for this blessing on us without which we could have not successfully our project.I am extremely grateful to **Dr.Madan.A. Sendhil, M.S., Ph.D.,** Chairman, Rathinam Group of Institutions, Coimbatore and **Dr. R.Manickam MCA., M.Phil., Ph.D.,** Secretary, Rathinam Group of Institutions, Coimbatore for giving me opportunity to study in this college.

I am extremely grateful to **Dr.R.Muralidharan, M.Sc., M.Phil., M.C.A., Ph.D.,** Principal Rathinam College of Arts and Science(Autonomous), Coimbatore.

Extend deep sense of valuation to **Mr.A.Uthiramoorthy, M.C.A., M.Phil., (Ph.D),** Rathinam College of Arts and Science (Autonomous) who has permitted to undergo the project.

Unequally I thank **Mr.K.Arunkumar, M.E., (Ph.D).,** Mentor and **Ms.V. Kanimozhi, M.E., (Ph.D).,** Project Coordinator, and all the Faculty members of the Department - iNurture Education Solution pvt ltd for their constructive suggestions, advice during the course of study.

I convey special thanks, to the supervisor **Mohammed harun Babu R** who offered their inestimable support, guidance, valuable suggestion, motivations, helps given for the completion of the project. I dedicated sincere respect to my parents for their moral motivation in completing the project.

List of Figures

List of Abbreviations

SSD	Single Shot Detector
CNN	Convolutional Neural Network
API	Application programming interface
R-CNN	Region Based Convolutional Neural Networks
YOLO	You Look Only Once
RPN	Region Proposel Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit
IOT	Internet of Things
PYTESST	Python Tesseract
RESNET	Residual Neural Network
I-CNN	Improved Convolutional Neural Network
GPU	Graphical User Interface

Abstract

Domain generation algorithms (DGAs) are used by attackers to generate a large number of pseudo-random domain names to connect to malicious command and control servers (CCs) . These domain names are used to evade domain based security detection and mitigation controls. Reverse engineering of malware samples to discover the DGA algorithm and seed to generate the list of domains is one of the techniques used to detect DGA domains. These domains are subsequently preregistered and sinkholed, or published on security device blacklists to mitigate malicious activity. This technique is time-consuming and can be easily circumvented by attackers and malware authors. Statistical analysis is also used to identify DGA domains over a time window, however many of these techniques need contextual information which is not easily or feasibly obtained. Existing studies have also demonstrated the use of traditional machine learning techniques to detect DGA domains. Our goal was to detect DGA domains on a per domain basis using the domain name only, with no additional information. T DGA classifier that leverages a Long short term memory based architecture for the detection of DGA domains without the need for contextual information or manually created features. We compared the performance of different LSTM based architectures by evaluating them against a dataset of 2 million plus domain names. The results indicated little difference in performance metrics among the LSTM architectures.

Chapter 1

Introduction

1.1 Cyber Security

Internet is one of the indispensable platform for each and every one in a routine past recent years for the activities such as entertainment, edutainment, it become a part of our life in a useful way meanwhile its harmful in all aspects. Recent day's malwares gain access through various medium while installing application, software, via mail etc. The prominent element on the internet is translating of domain names to IP addresses and mapping it for internet users using Domain name system [7]. This elementary access through internet portal increase traffic over the DNS and provides access to the attackers to gain information easily.

Botmasters use botnets for all types of malicious access like stealing information, data from the victim system without his knowledge and permission. Additionally ransomware attacks are distributed by attack DoS attack (denial-of-service) through the victim's system or encrypting the victims drive by connecting CC center. This communication serves multiple functions at once and [14] malware will gain access such as passwords access credentials etc. Botmasters can easily establish such a communication to the CC server with an IP address by passing the malware through Domain generation algorithm (DGA).

DGA generally generates wide range of random domain names at a time. In this randomly generated domains malware males a attempt to connect with the DNS operator server. If the domain is been easily connect to the DNS, botmaster can easily pass through the network by passing malware [13]. The malware can acquire IP for the registered domain and can easily communicate with the CC server. To detect, DGA's blacklisting method was used [9]. Blacklisting methods fails in detecting newly generated DGA domain names. To handle new types of DGA generated domain, machine learning methods were introduced [11], but machine learning completely depends on feature extraction and domain knowledge also needed to perform the task. Moreover, classical machine learning based classifier can be easily broken by attackers in a conflict environment. Nowadays, deep learning architectures with character embedding is used for DGA detection and as well as categorization [7], [7], [19]. These deep learning based architectures vulnerable to imbalanced DGAs. To handle imbalanced DGAs, in this work Cost-Sensitive long short-term memory (CS-LSTM) is proposed as a novel method for DGA categorization.

1.1.1 Domain Name System (DNS)

Domain Name System (DNS) translate domains names to IP addresses in a prominent way. All domain names of the website are possessed with corresponding IP addresses. These IP addresses helps in identification of websites while surfing, all these are numerical data which is not easy to remember so for each IP corresponding domain names are generated, in which DNS act as a collective form of database to store domain names and IP addresses. DNS has hierarchy in which domain names are comprised. The hierarchy flow consists of Root level, Top level domains, second level domains Subdomains and host as shown in Figure 1.1.

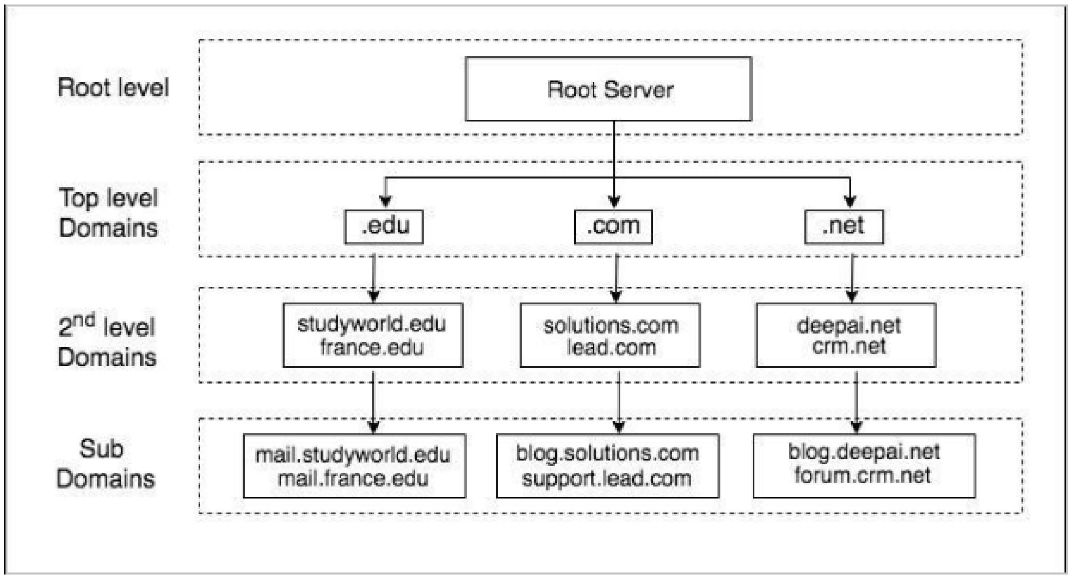

Figure 1.1: Hire-achy of Domain name system (DNS)

1.1.2 Domain fluxing and domain generation algorithms

Flux usually refers to something constantly changing, In the case of domain fluxing bots use Domain generation algorithm to produce thousands of domain names randomly, which is registered by the botnet operator[17,18,19]. All bots will send queries to DNS randomly until they resolve the address in the CC server as shown in Figure 1.2.

These makes security and administrators difficult to shut down the botnets from CC serve. To overcome such issues our work relies on various machine learning models to evaluate the ability of DGA families' detection.

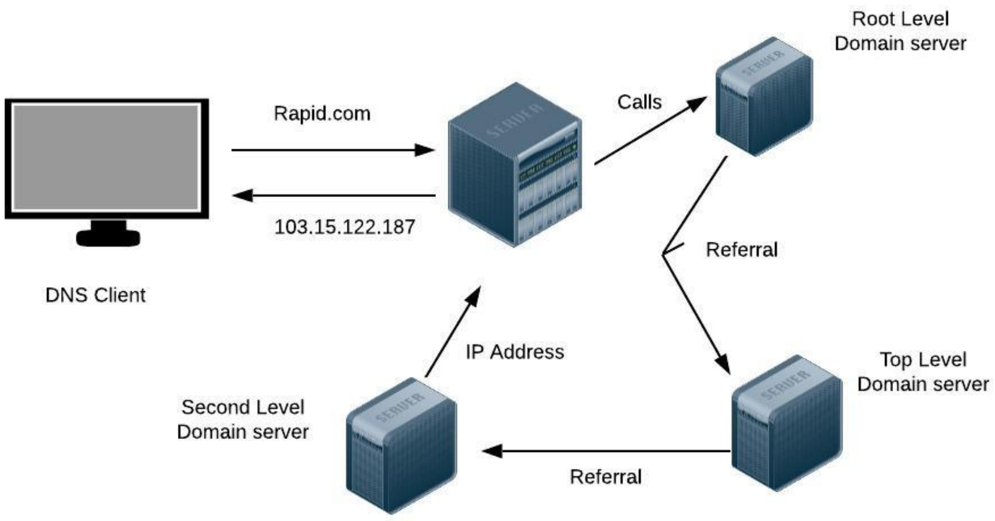

Figure 1.2: Resolving of DGA-Generated domains

1.1.3 Botnet

A bot-net is a major threat to cyber security applications. A Bot-net is a sub-net of under-controlled remote machines typically called bot-master. Bot performs variety of task on receiving commands from the bot-master such as email scam, steeling data without user knowledge, credit card scam etc. Bot are first send through bot-master by a malware and affects the system, these happens through email attachment or while accessing social media posts. These bots create a software vulnerability from backdoor access of the system. Once the malware take control of the system it use Internet as a medium to connect with C C server. Unless or until user doesn't knows that system is being under bot-net control. Once the bot-master has a need to access the system of the user, bot-net acquires what they need from the affected system as shown in figure 1.3.

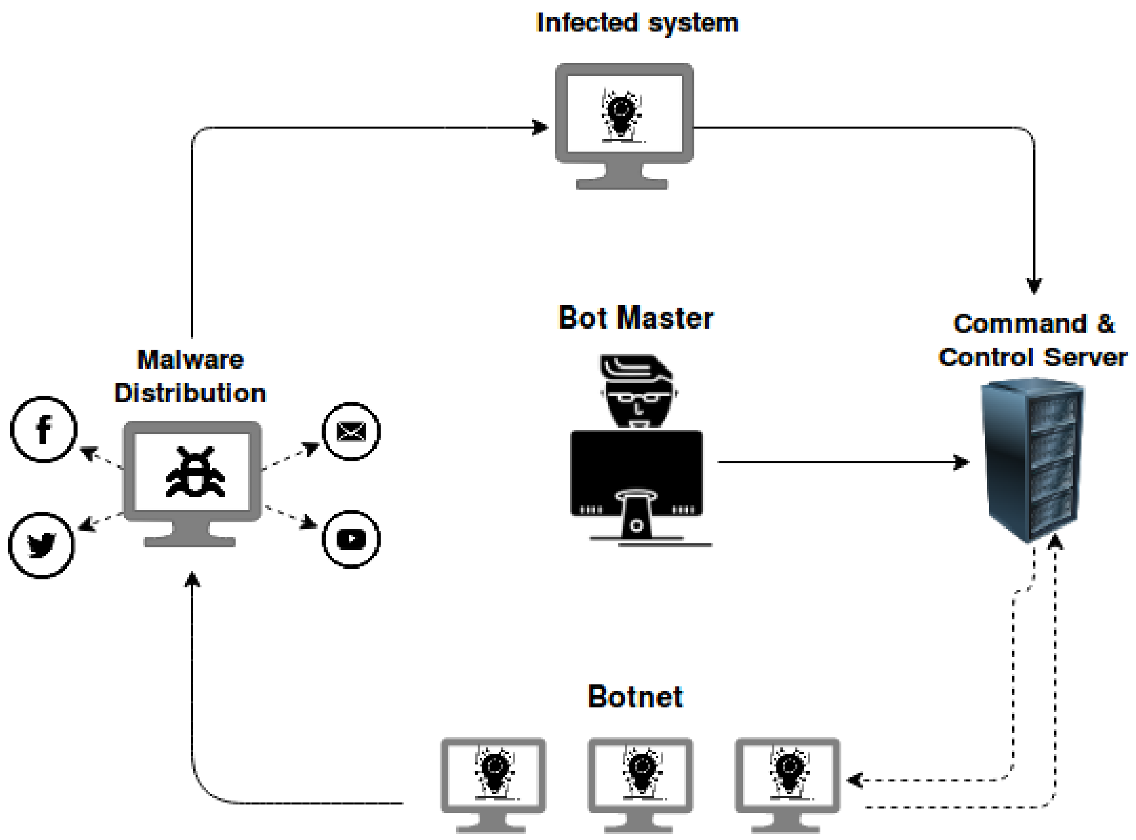

Figure 1.3: Typical infrastructure of Botnet

1.2 Literature Survey

Kührer, M., Rossow, C. and Holz [1] blacklisting are used in system protection from malwares. In this an emprical approach on a publically available data-set with 15 malware and 4 blacklist content listed data is analyzed. The main aim is to classify blacklist to understand the nature of the listed domains according to corresponding IP addresses. As per experimentation parked domains are identified, graphical approach is determined to list the sinkholes.

Antonakakis, Manos [2] proposed a technique to identification of domain names without reverse engineering process, work mainly focuses on the NX-Domain names. Clustering approach is taken and domains are classified into clusters based on their characters tics features. This detected method is named as Pleiades, which is comprised to monitor the traffic flow along the recursive server.

Yadav, Sandeep [3] explained the DNS traffic along the network, analysis of patterns in which domain flux is happening is studied. The distribution pattern of a particular numerical character is taken and mapped to extract the information using bigram techniques. Distance is calculated using KL and Edit distance method, the output is mapped to different IPv models to consider the data traffic along the internet service provider.

Antonakakis, Manos [4] a novel approach called Notos a DNS dynamic reputation system, these approach helps in calculate score by analyzing the transverse edge of the networks. All the information are extracted from different resoueces of DNS names, based on the information clustering is done to identify the behaviour of malicious and legitimated. This method comes up with 96.8% of true positve rate and with loss of 0.38% for identifying DNS traffics.

Bilge, Leyla ,[5] a new method EXPOSURE is used analysis DNS domains at large scale, in this approach 15 fetures are extracted and characterized in basis of DNS queries and serves a command to C & C server for specific threat analysis.

Yu, Bin, Jie Pan, Jiaming Hu, [7] proposed various deep learning algorithm for DGA detection. Nearly about 2M domain names is used for evaluation to classify domain names either benign or malicious. Five deep learning architecture such as RNN architecture in which Endgame and CMU model used, In CNN architecture NYU and Invincea model used, In hybrid RNN/CNN architecture MIT model is used. In which all network perform equally well of 97-98% of accuracy with false positive rate of 0.001. Using hang craft feature Random forest classifier achieve 91.57% with false positive rate of 0.001.

Zeng, F., Chang, S,[8] focused on CNN architecture such as VGG net, Alex net, Squeeze net, inception, Res net used to classify DGA names. Transfer learning approach is used to extract features from raw input, using various network about 99.86% with false positive rate of 0.011 is achieved.

Yu, B., Gray, D.L., Pan, J., De Cock, [9] Supervised learning approach used for DGA detection. Various filtering steps are used to obtain real time data samples from DNS servers. A comparison study is done between LSTM and CNN network in which LSTM performs with high performance rate.

Vinayakumar, R., Prabaharan Poornachandran [18] performed several analysis work on large scale to give a detail of cyber activity on DNS server and how to detect malware in early stage. For analysis deep learning methods used by analyzing the DNS information at Tier-1 Internet service provider. It is stated that the developed framework can detect nearly

2million malicious activities in real time and can give real time warning.

Mac, H., Tran, D., Tong, V., Nguyen, L.G,[10] investigated various handcrafted features such as Hidden markov model, Decision tree, SVM (Support vector machine) and C4.5 used. In feature based technique entropy and length are consider with equivalent dictionary score followed by n-gram normality score for the domain names. Deep learning network such as CNN-LSTM, Bi-directional LSTM used, from which SVM and Bidirectional LSTM achieve higher classification rate on both binary and multi-class dataset.

Woodbridge, J., Anderson, H.S., Ahuja, [13] focused LSTM architecture to classify domain names, this paper helps to understand the data distribution. Using data distribution a characteristic of DGA family is analyzed by taking the sequence and length features for each domain names separately. Using LSTM model the sequence characteristic of the domain names are absorbed and it helps in classifies the domain name according to corresponding classes.

Vinayakumar, R., Soman, K.P. and Poornachandran P [11] collected DNS logs from server, these DNS log details are used to classify DGA families. Author mainly focuses on Deep learning architecture and data collection for cyber security application. As per analysis LSTM approach gives a higher detection rate on the collected data-set.In [14] this paper author used nearly one million DGA domain names from various resources such as Open DNS, Alexa data-set, OSNIT data which consists of 17 DGA malware families. Author consider the effectiveness these data on various in various deep learning architectures at a large scale approach

Curtin, Ryan R., Andrew B. Gardner[15] analyzed and give a solution to measure the complexity of distribution of DGA families using a technique called smash-word score. These smash-word score help in identify the resemblance of DGA families based on English words , considering the smash- word count machine learning model and deep learning models are built. [17] proposed various feature extraction methods used by various research work using machine learning algorithms. This paper state the state of art of all proposed work and how feature extraction methods contributes in binary and multi-class problems

Mohan, Vysakh S., R. Vinayakumar [16] proposed a new approach S.P.O.O.F Net, combination of a Convolution Neural Network (CNN) and LSTM which is an embedding concepts from natural language processing (NLP) is been embedded into cybersecurity use cases. The proposed model is incorporated with feature engineering method Bi-gram and conventional CNN with character level embedding. The SPOOF Net model comes with accuracy scores of 98. 3% for DGA detection and 99% for malicious URL detection.

1.3 Objectives

- Perform a study of the Domain generation algorithm (DGA) by analyzing 21 DGA based malware families and its variants
- Handle Multi-class class imbalance along data using Cost-Sensitive LSTM approach.

1.4 Thesis Overview

Chapter 2 describes background theory required for the proposed methods and chapter 3 presents the proposed method. Chapter 4 describes the experimental setup, data-sets and the baseline approaches used in the present work. Chapter 5 reports the experimental results and discussion and the conclusion derived from the present work is given in Chapter 6.

Chapter 2

Background

2.1 Recurrent Neural Network - RNN

In neural network system recurrent nets are designed to get the sequence patterns of the data. The major difference between feed food network from recurrent neural network is time is being introduced in particular and output is been fed back in a loop order. We have a simple RNN network with four input nodes as shown in figure

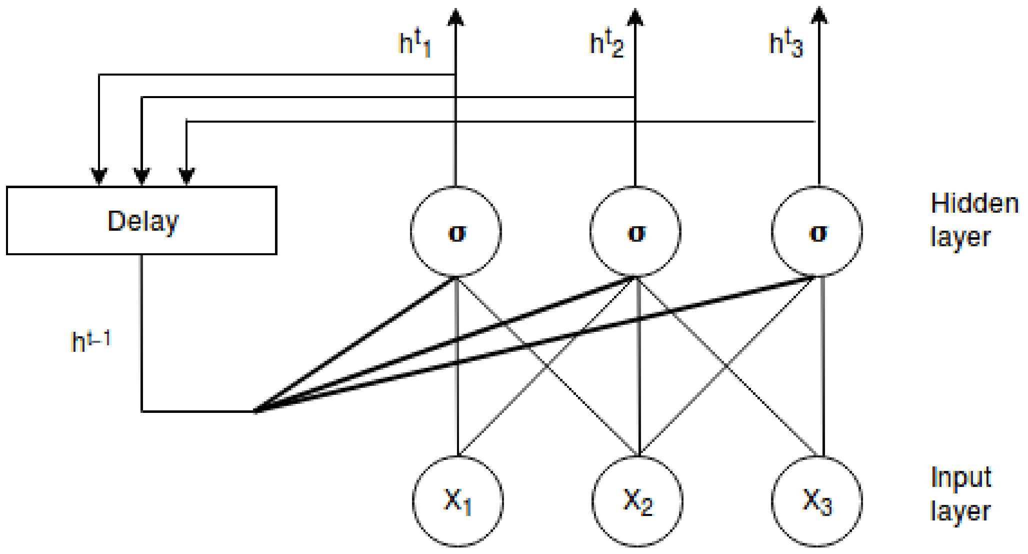

Figure 2.1: Recurrent neural network

The input nodes are nourished into the hidden layers with sigmoid activation function and the output is fed into the same hidden layer. From this it is identified that the output the hidden layer is been passed through the delay block and give access to the h_{t-1}input to the hidden layer, it simply states that RNN network is time dependent. In practice RNN are not used often. The main problem faced by RNN is vanishing gradient issue. Ideally, all neural network needs long memory space to learn the relation between the each sentence in a time depended order. Let us consider RNN is represented as :

$$h_t = \sigma(Rx_t + Jh_{t-1}) \tag{2.1}$$

Where R and J are weight matrix which connects the recurrent inputs to the recurrent output. Activation function softmax is performed on outputs h_t. Let us assume for three RNN time steps as shown in equation 2

$$h_t = \sigma(Rx_t + J(\sigma(Rx_{t-1} + J(\sigma(Rx_{t-2}) \tag{2.2}$$

From the equation it clearly states that when layers are more deep along the network,gradient error occurs with respect to the weight matrix R through time during back-propagation along the neurons.

$$\frac{E_3}{\partial R} = \frac{\partial E_3}{\partial out_3}\frac{\partial out_3}{\partial E_3}\frac{h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_R} \tag{2.3}$$

A approximation on whats happening in back-propagation is shown in equation, in which each one of these is helpful in calculating sigmoid function gradient value. Problem occurs in sigmoid function is rounding off of values when the output gets close to 1 or 0. When the gradient values multiplies many values in the neuron there is a potential difference in the output and leads to vanishing gradient $\frac{\partial E}{\partial R}$.RNN posses with vanishing gradient problem, it is overcome by using Long short term memory LSTM network

2.2 Long Short-Term Memory - LSTM

. LSTM basically poss ed with memory blocks, simply it creates an input memory block which reduces the smaller gradient effect, and previous inputs which are fed into the input layer is controlled by forget gate and make LSTM ahead from the vanishing gradient issue, forget gate used to determine the state of input whether its remembered or forgot-ted. Totally LSTM has three gates input gate, output gate and forgot gate.

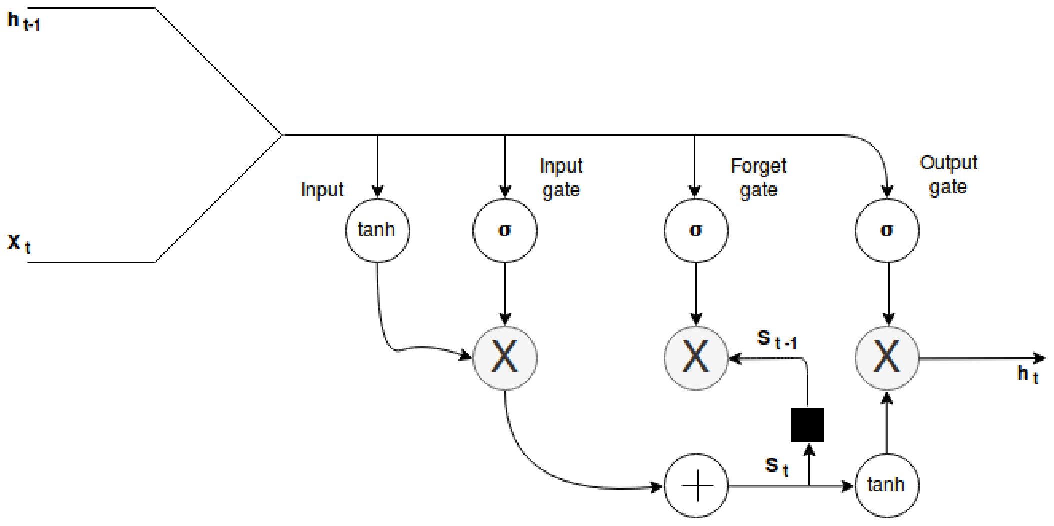

Figure 2.2: Long short term memory

The entire flow of LSTM is been shown in fig, x_t as input, h_{t-1} as a output cell. In this activation function *tanh* is take and distinguishing values are from -1 to 1. The following theroy is expressed as

$$g = tanh(b^g + x_t R^g + h_{t-1} J^g) \tag{2.4}$$

Where weight from the previous cell input and output is denoted by R^g and J^g. Bias is denoted by b^g, g is the input weight of the neuron.

The output of the input gate undergoes element wise multiplication, with sigmoid nodes with x_t of weights and input value ofh_{t-1} . The input gate of LSTM is expressed as

$$k = \sigma(b^k + X_t U^k + h_{t-1} V^k \tag{2.5}$$

The output of the LSTM cell is expressed as where $\circ$ is multiplication operator element wise.

$$g \circ k \tag{2.6}$$

In the forget gate a new state is introduce S_t . This inner state S_t provides a internal loop with time step adding to the $g \circ k$ input state. Forget gate usually set a node for activation function which is S_{t-1}. It Remembers the information from the previous input state, this helps LSTM to learn the exact context fed into the network. The Forget gate is expressed as

$$l = \sigma(b^1 + X_t U^1 + h_{t-1} V^1) \tag{2.7}$$

S_{t-1} is output of the forget gate and each time the inputs added to the this gate, all inputs are filtered without multiplication it is mixed along with sigmod function and weights. This helps in eliminating vanishing gradient problem. Finally the output gate posses with gating function from each cell and produce the output.

Chapter 3

Methodology

3.1 Dataset Description

The data used to train the classifier is taken from the CSE-CIC-IDS2018 dataset provided by the Canadian Institute for Cybersecurity. It was created by capturing all network traffic during ten days of operation inside a controlled network environment on AWS where realistic background traffic and different attack scenarios were conducted. As a result the dataset contains both benign network traffic as well as captures of the most common network attacks. The dataset is comprised of the raw network captures in pcap format as well as csv files created by using CICFlowMeter-V3 containing 80 statistical features of the individual network flows combined with their corresponding labels. A network flow is defined as an aggregation of interrelated network packets identified by the following properties:

Source IP Destination IP Source port Destination port Protocol The dataset contains approximately 16 million individual network flows and covers the following attack scenarios:

Brute Force DoS, DDos Heartbleed, Web Attack, Infiltration, Botnet

In this work 3 different methods are obtained for experimental analysis.

3.2 Proposed Method

1. Method 1: Standard Long- short term memory

2. Method 2: LSTM + Glove embedding

3.3 Character level encoding

Proposed architecture consists of input layer followed by embedding layer where character level features are extracted and passed. The output of the embedding layer ispossessed with LSTM layer with softmax activation function followed by output layer as shown in figure

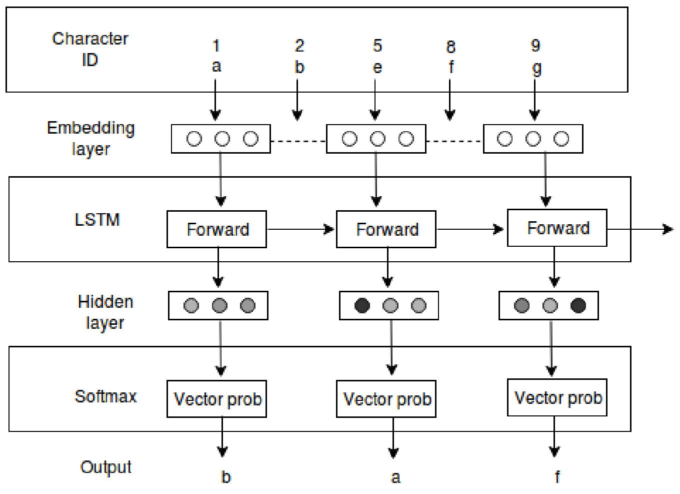

Figure 3.1: Character Level Encoding

Domain name representation is typically referred as encoding. The encoding of the domain name consists of two steps. The raw domain names are pr-processed into characters in the first step. In pre-processing , the top-level domains is removed and all characters converted into lower case. Second step involves creating vocabulary with only initial step of using the training data. The parameter value of vocabulary size is based on the symmetry between each class training vectors and number to be learned for the given task. Here only the characters that meet the minimum frequency is selected to limit the size of the vocabulary. Followed by the initial step pf creating vocabulary, each character is assigned to a unique ID and each unique ID is a vector denoting the size if the vocabulary D. Default key value 0 is assigned to the unknown characters, and using look up-table operation, the unique characters are transformed into feature vectors.

3.4 Glove Embedding

Pre-trained embedding of words is a key element in NLP's deep learning. In mainstream deep learning, the pioneer of word embedding is the famous word2vec. Glove is a pre-trained embedding technique,In fact, GloVe is a much more principled approach to embedding words, which generally offers profound insights into embedding words. The main aim of glove is

1) Meaning of a words is captured and mapped into a vector space

2) Glove embedding helps in reduction of loss by updating the weights.

The underlying principle behind GloVe can be stated as follows: the co-occurrence ratios between two words in a context are strongly connected to meaning.

This sounds difficult but the idea is really simple. Take the words "ice" and "steam", for instance. Ice and steam differ in their state but are the same in that they are both forms of water. Therefore, we would expect words related to water (like "water" and "wet") to appear equally in the context of "ice" and "steam". In contrast, words like "cold" and "solid" would probably appear near "ice" but would not appear near "steam".

In GloVe, principled based approach is performed. The first step is to build a co-occurrence matrix. GloVe also takes local context into account by computing the co-occurrence matrix using a fixed window size (words are deemed to co-occur when they appear together within a fixed window). For instance, the sentence

The probabilities shown here are basically just counts of how often the word k appears when the words “ice” and “steam” are in the context, where k refers to the words “solid”, “gas”, “water”, and “fashion”. As you can see, words that are related to the nature of “ice” and “steam” (“solid” and “gas” respectively) occur far more often with their corresponding words that the non-corresponding word. In contrast, words like “water” and “fashion” which are not particularly related to either have a probability ratio near 1. Note that the probability ratios can be computed easily using the co-occurrence matrix. The detail overview of glove embedding is shown in figure 3.2

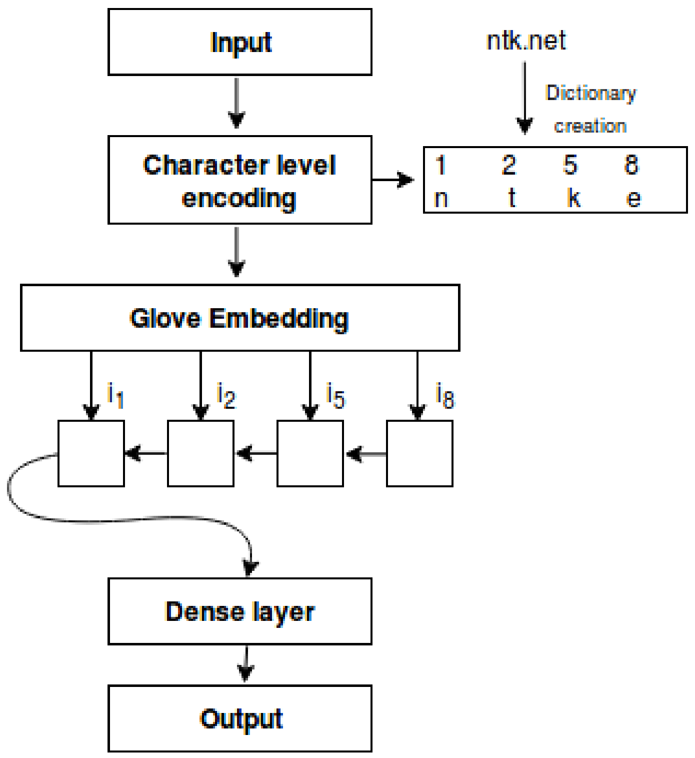

Figure 3.2: Glove embedding architecture

Chapter 4

Results and Discussions

4.1 Experimentation 1

Deep layer algorithm such as RNN, LSTM, CNN, GRU is used. To evaluate the performance of all algorithms constant network parameter is set for all algorithm such as learning rate as 0.1, batch size as 32, and dropoutas 0.2. In recurrent algorithm RNN, LSTM, methods are adopted, in which softmax is used as a activation function with embedding vector length of 128and maximum feature of 40, in LSTM and GRU memory cell size is set as 128 .

Algorithms	**Accuracy (%)**	**Precision**	**Recall**	**F1- Score**
GRU	0.65	0.62	0.67	0.66
CNN	0.70	0.68	0.69	0.61
RNN	0.72	0.67	0.74	0.69
LSTM	0.78	0.71	0.79	0.76

Table 4.1: Summary of test results on different deep learning architecture

In convolution architecture 1D convolution is adopted for analysis, with 64 filter of length3. .By evaluating the model performance of each moreover all network are differs with high amount of variation of accuracy from which GRU not performed well compared to other algorithms and LSTM outperformed with 78.2% accuracy as show in table 1 as per experimentation1.

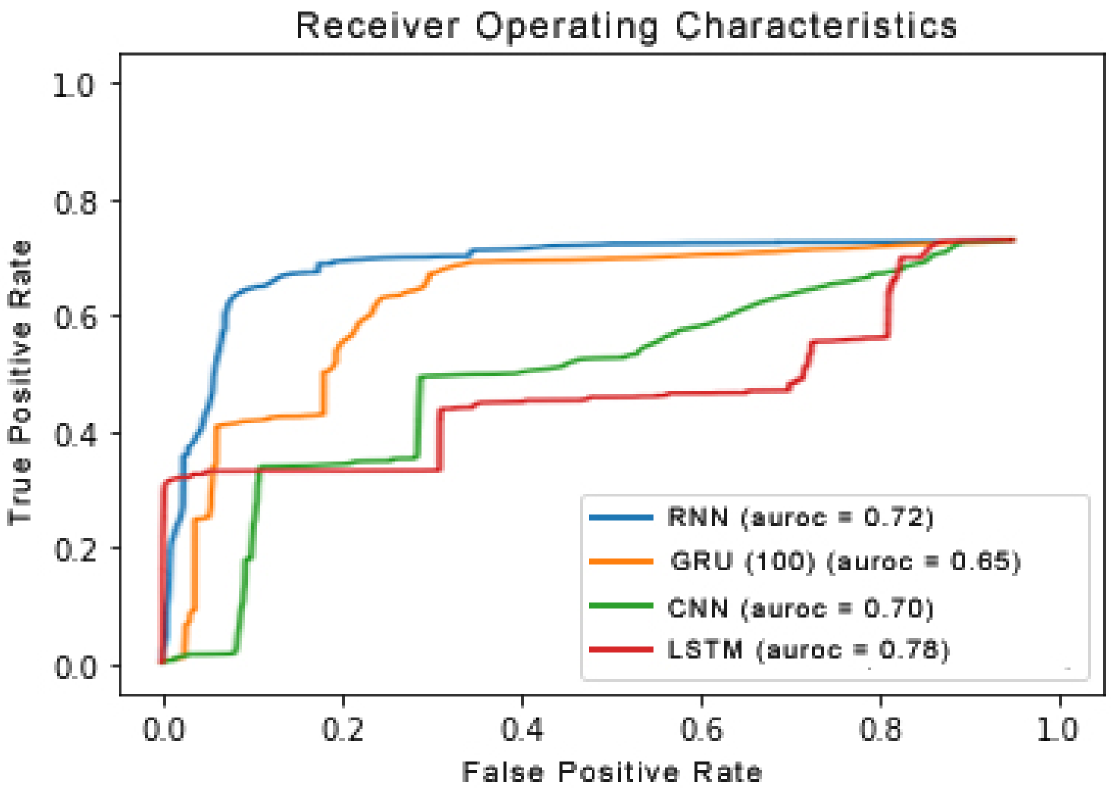

Figure 4.1: Receiver operating characteristic (ROC) curve of RNN, GRU, CNN, LSTM

To get a detailed overview of the correctly predicted and non-predicted data confusion matrix is drawn as show in figure 4.2

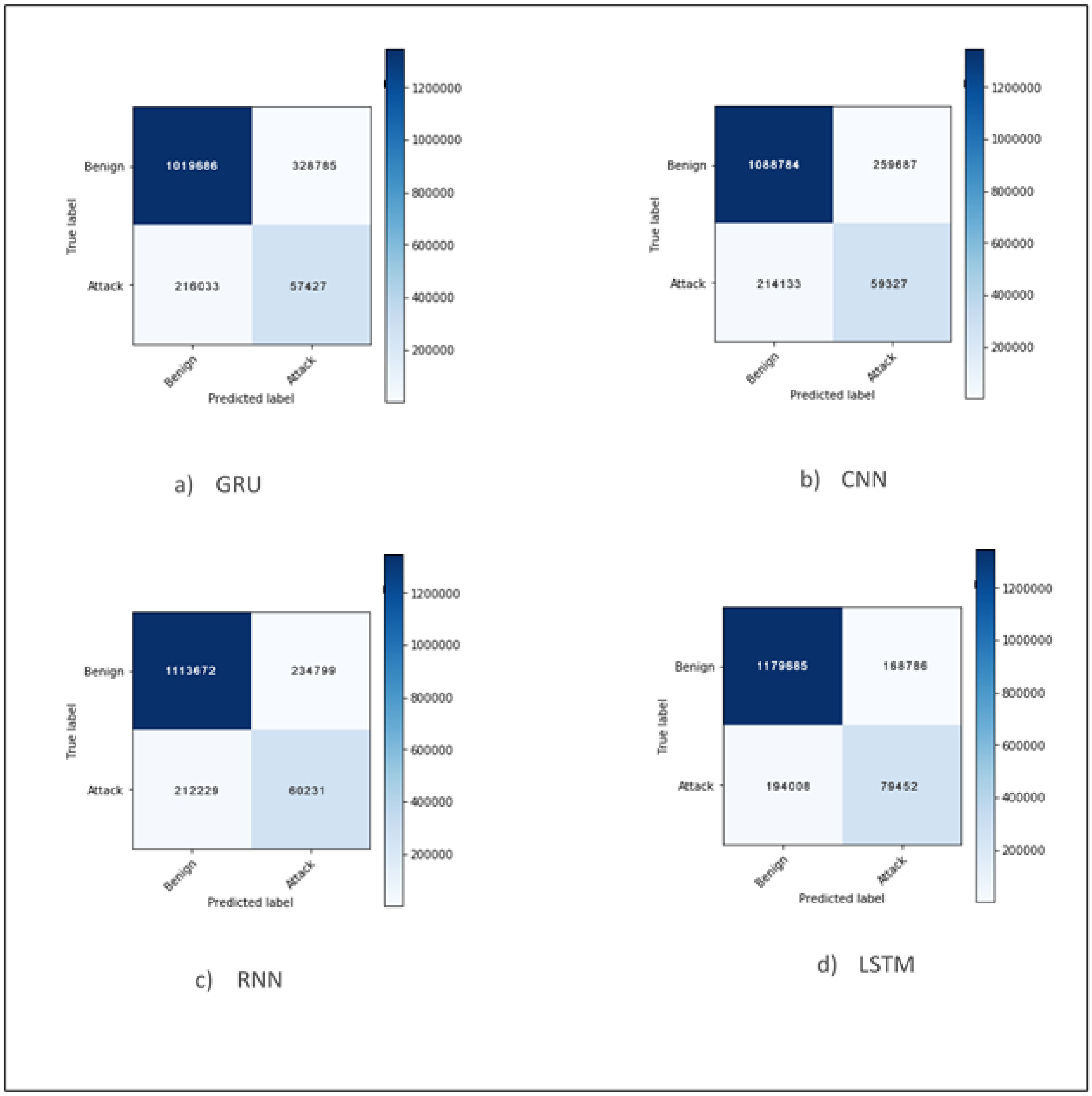

Figure 4.2: Confusion matrix for GRU,CNN,LSTM, RNN (Experimentation -1)

4.2 Experimentation 2

On account of LSTM performance in trial 2 experimentation hyper-parameter tuning is applied with various tuning parameters in LSTM architecture. In trail 2 embed-ding vector length is been kept constant of 50, and various LSTM memory size is used such as 256, 128, 64 and 32 to identify the actual parameter for analysis. It is clearly state that when number of input neuron decreases accuracy increases as per result input neuron with 64 and 32 is performed with good result of 84 and 87%

Algorithms	Embedding	LSTM	Accuracy (%)	Precision	Recall	F1- Score
LSTM	50	256	0.74	0.79	0.74	0.71
LSTM	50	128	0.79	0.78	0.81	0.76
LSTM	50	64	0.84	0.89	0.81	0.82
LSTM	50	32	0.87	0.83	0.84	0.88

Table 4.2: Summary of test results on LSTM with different LSTM Size

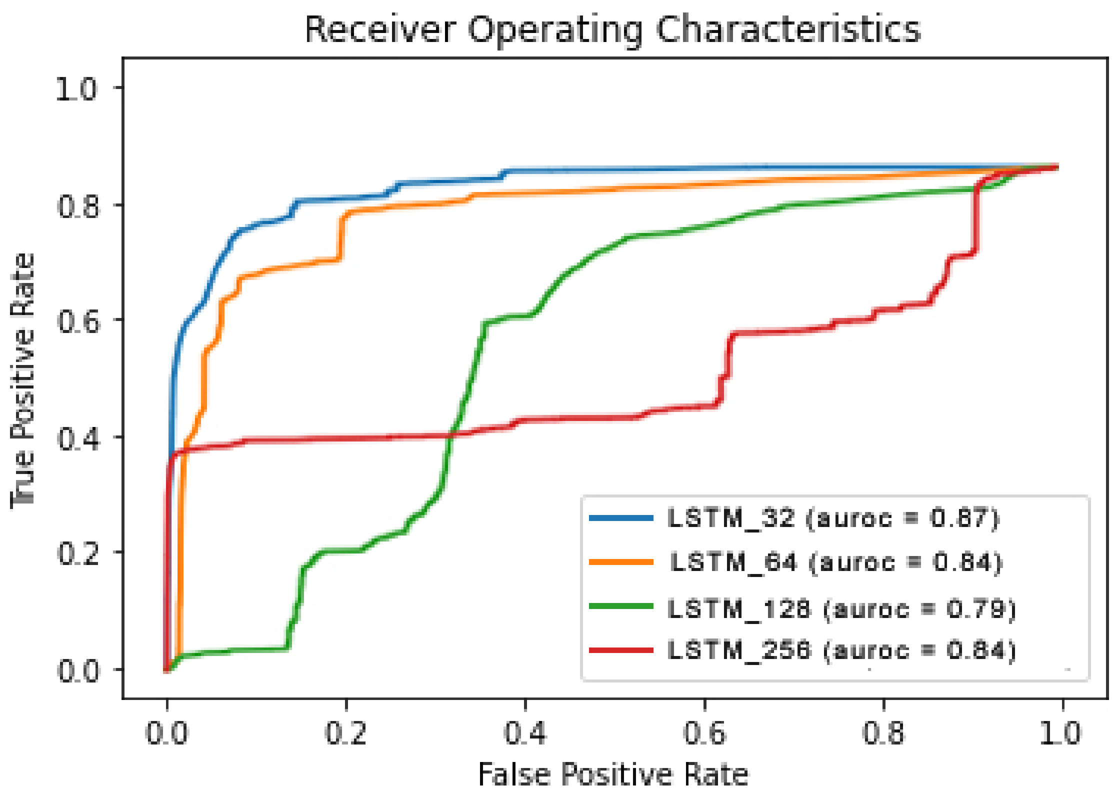

Figure 4.3: ROC curve of LSTM32, LSTM64, LSTM128, LSTM256

To get a detailed overview of the correctly predicted and non-predicted data confusion matrix is drawn as show in figure

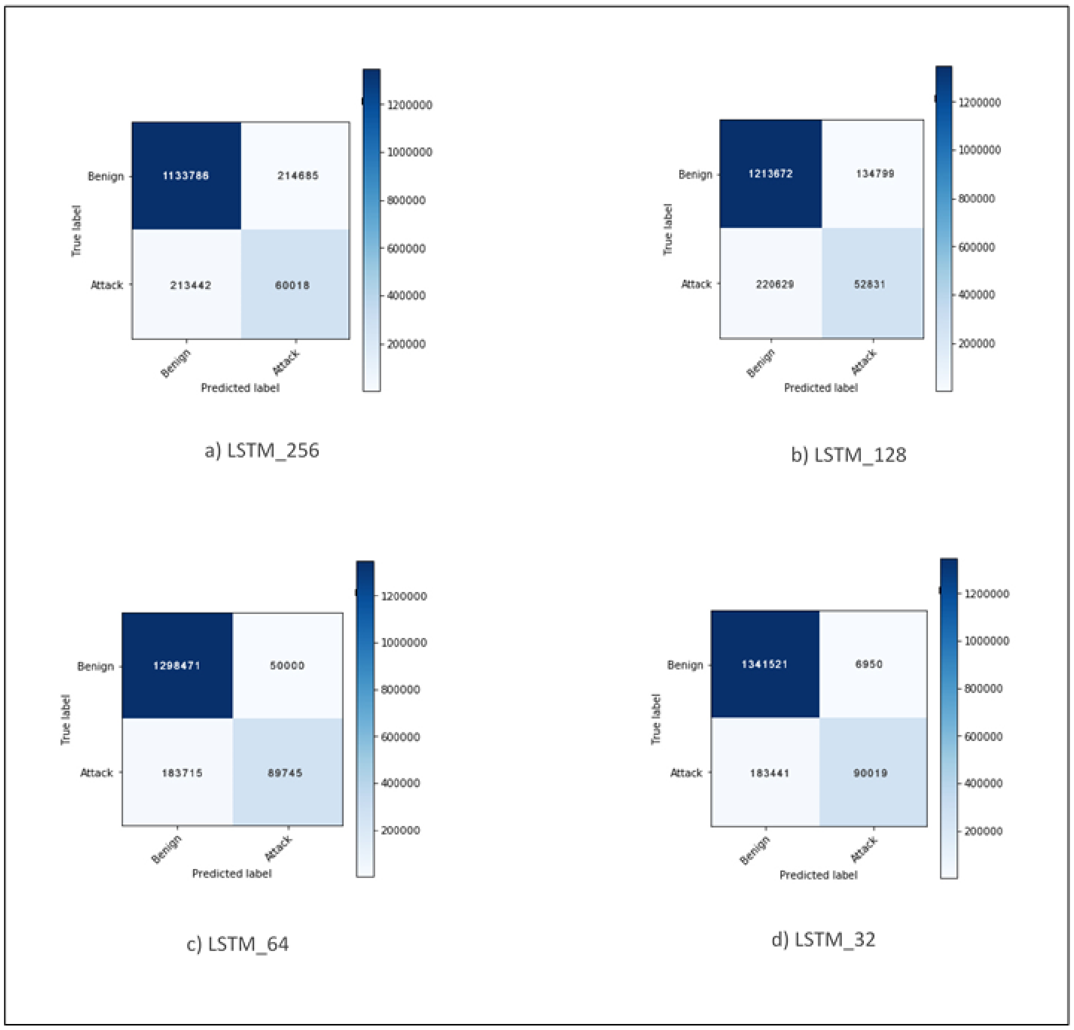

Figure 4.4: Confusion matrix for LSTM32, LSTM64, LSTM128, LSTM$_2$56$(Exp2)$

LSTM of size 64 and 32 gives a better result compared to previous experimentation only benign classes are predicted almost of 93 %. Attack classes are not predicted correctly with high amount of misclassification which in accuracy declination. While analyzing the data it clearly states that there are more amount of trainable parameters are there compared with attack class which result in miss classification And decrease in accuracy . To over come this misclassification and for insufficient data glove embedding technique is applied to increase the weights of the attack class . Glove embedding is performed for LSTM of input size 64 and 32 .

4.3 Experimentation 3

On account of LSTM performance in trial 3 experimentation LSTM with glove tuning is applied with various tuning parameters in LSTM architecture various LSTM memory size is used such as 256, 128, 64 and 32 similar to experiment 2 to see how glove embedding performs on different memory size. It is clearly state that when number of input neuron decreases accuracy increases as per result input neuron with 64 and 32 is performed with good result of 93 and 97% . Compared to experimentation 2 all LSTM size of 256, 128, 64 ,32 performs well .

Algorithms	Embedding	LSTM	Accuracy (%)	Precision	Recall	F1- Score
LSTM	50	256	0.74	0.79	0.74	0.71
LSTM	50	128	0.79	0.78	0.81	0.76
LSTM	50	64	0.84	0.89	0.81	0.82
LSTM	50	32	0.87	0.83	0.84	0.88

Table 4.3: Summary of test results on LSTM with different LSTM Size

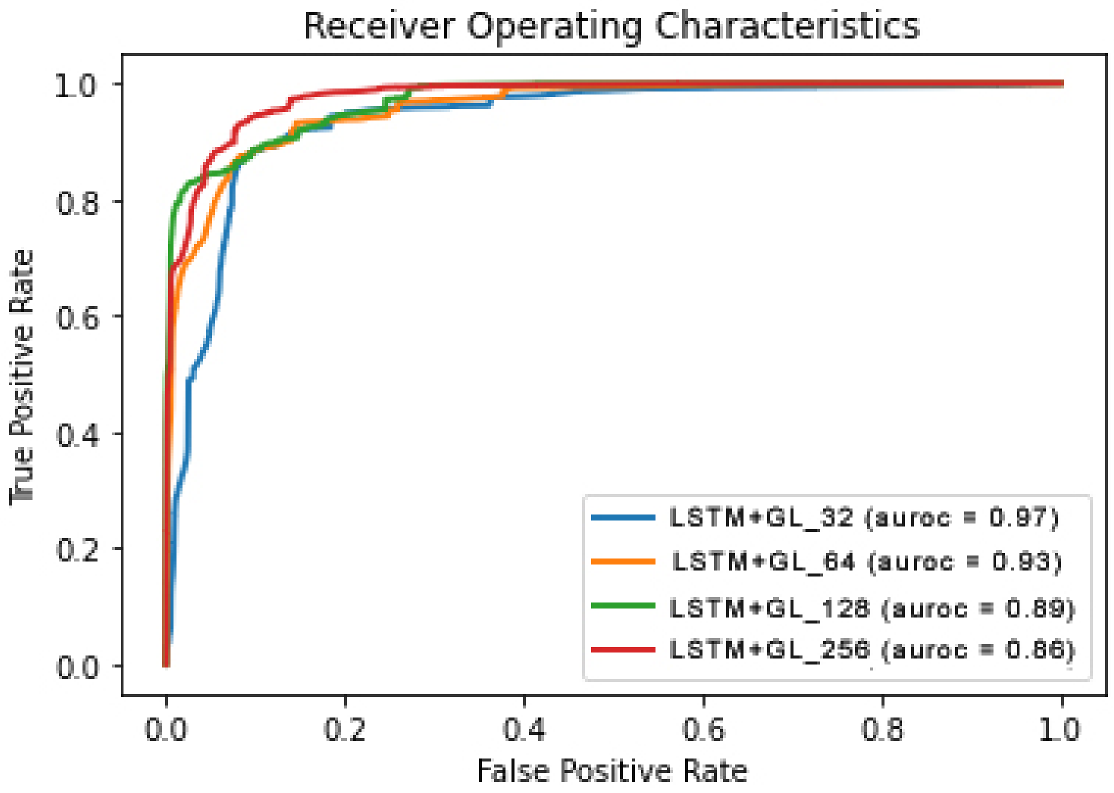

Figure 4.5: ROC of LSTM+GL32, LSTM GL 64, LSTM GL 128, LSTM GL 256

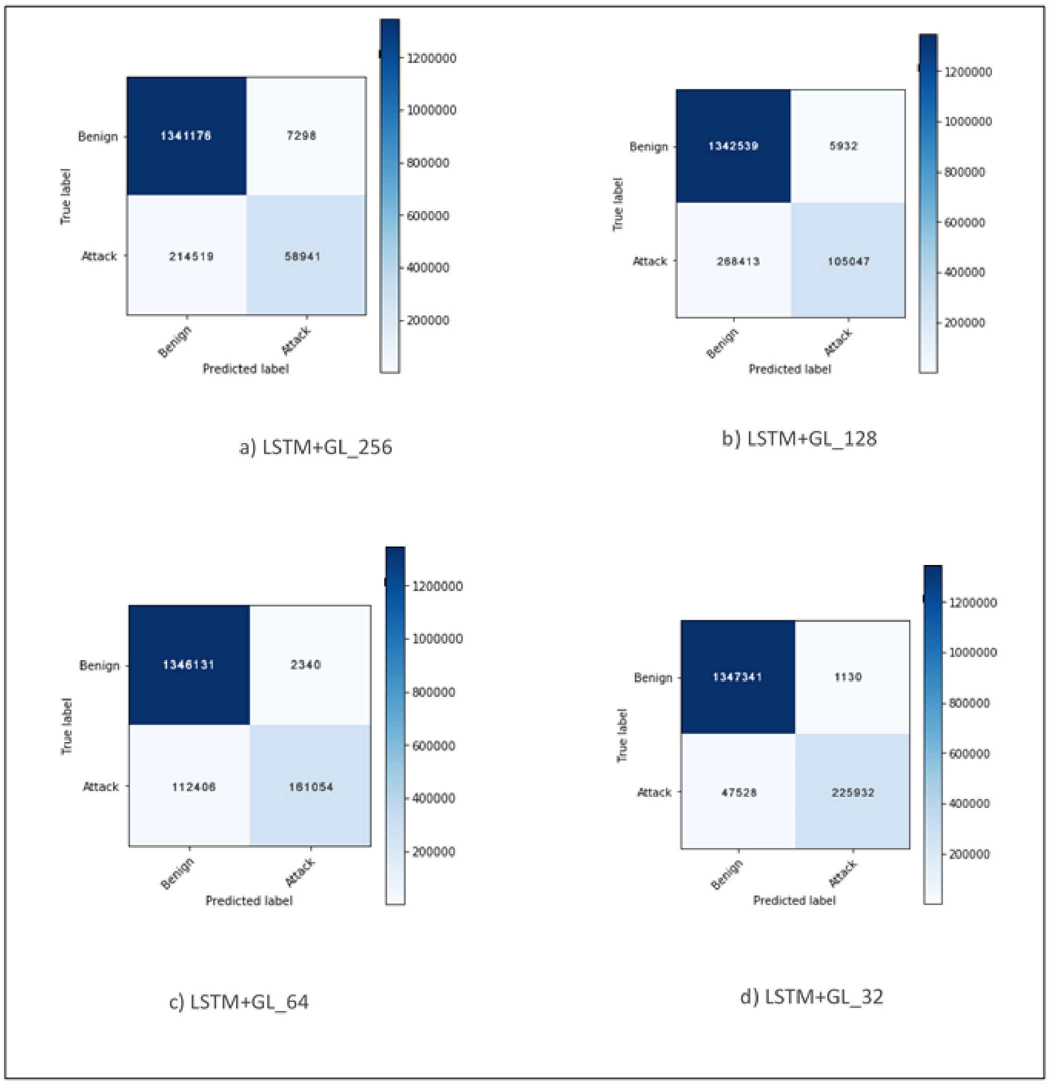

Figure 4.6: Confusion matrix for LSTM+GL32, LSTM GL64, LSTM GL 128, LSTM GL 256 (Exp-3)

Chapter 5

Conclusion

This article provided an approach to classifying domains produced by DGA using LSTM networks. LSTMs are advantageous compared to other methods because they use raw domain names as their input and there is no need to manually generate characteristics that are hard to keep and in an adversarial machine learning environment.In addition DGA dataset posses with class imbalance, to overcome this imbalance of data a new approach Glove-LSTM is applied on DGA Family categorization. Cost-Sensitive helps in understand the importance of each class and classify the classes to a particular class,Glove embedding method is used to increase the performance of GL-LSTM. The proposed method performs well and obtained better result in multiclass DGA dataset. Further the proposed method can be used to handle imbalanced dataset.

References

1. Kührer, Marc, Christian Rossow, and Thorsten Holz. *Paint it black: Evaluating the effectiveness of malware blacklists.* In International Workshop on Recent Advances in Intrusion Detection, pp. 1-21. Springer, Cham, 2014.

2. Antonakakis, Manos, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. *From throw-away traffic to bots: detecting the rise of DGA-based malware.* In Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), pp. 491-506. 2012.

3. Yadav, Sandeep, Ashwath Kumar Krishna Reddy, A. L. Reddy, and Supranamaya Ranjan. *Detecting algorithmically generated malicious domain names.* In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 48-61. ACM, 2010.

4. Antonakakis, Manos, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. *Building a dynamic reputation system for dns.* In USENIX security symposium, pp. 273-290. 2010.

5. Bilge, Leyla, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. "EXPOSURE:

Finding Malicious Domains Using Passive DNS Analysis." In Ndss, pp. 1-17. 2011.

6. Attardi, Giuseppe, and Daniele Sartiano. *Bidirectional LSTM Models for DGA Classification.* In International Symposium on Security in Computing and Communication, pp. 687-694. Springer, Singapore, 2018.

7. Yu, Bin, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. *Character level based detection of DGA domain names.* In 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2018.

8. Zeng, Feng, Shuo Chang, and Xiaochuan Wan. *Classification for DGA-based malicious domain names with deep learning architectures.* In 2017 Second International Conference on Applied Mathematics and information technology, p. 5. 2017.

9. Yu, Bin, Daniel L. Gray, Jie Pan, Martine De Cock, and Anderson CA Nascimento. *Inline DGA detection with deep networks.* In 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 683-692. IEEE, 2017.

10. Mac, H., Tran, D., Tong, V., Nguyen, L.G. and Tran, H.A., 2017, December. *DGA botnet detection using supervised learning methods.* In Proceedings of the Eighth International Symposium on Information and Communication Technology (pp. 211-218). ACM.

11. Vinayakumar, R., Soman, K.P., Poornachandran, P. and Sachin Kumar, S., 2018. Evaluating deep learning approaches to characterize and classify the DGAs at scale. Journal of Intelligent Fuzzy Systems, 34(3), pp.1265-1276.

12. Lison, Pierre, and Vasileios Mavroeidis. *Automatic detection of malware-generated domains with recurrent neural models.* arXiv preprint arXiv:1709.07102 (2017).

13. Woodbridge, Jonathan, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. *Predicting domain generation algorithms with long short-term memory networks.* arXiv preprint arXiv:1611.00791 (2016).

14. Vinayakumar, R., Soman, K.P. and Poornachandran, P., 2018. *Detecting malicious domain names using deep learning approaches at scale.* Journal of Intelligent Fuzzy Systems, 34(3), pp.1355-1367.

15. Curtin, Ryan R., Andrew B. Gardner, Slawomir Grzonkowski, Alexey Kleymenov, and Alejandro Mosquera. *Detecting DGA domains with recurrent neural networks and side information.* arXiv preprint arXiv:1810.02023 (2018).

16. Mohan, Vysakh S., R. Vinayakumar, K. P. Soman, and Prabaharan Poornachandran. *SPOOF net: syntactic patterns for identification of ominous online factors.* In 2018 IEEE Security and Privacy Workshops (SPW), pp. 258-263. IEEE, 2018.

17. Kwon, Jonghoon, Jehyun Lee, Heejo Lee, and Adrian Perrig. *PsyBoG: A scalable botnet detection method for large-scale DNS traffic.* Computer Networks 97 (2016): 48-73.

18. Vinayakumar, R., Prabaharan Poornachandran, and K. P. Soman. *Scalable framework*

for cyber threat situational awareness based on domain name systems data analysis. In Big Data in Engineering Applications, pp. 113-142. Springer, Singapore, 2018.

19. Sun, Y., Kamel, M.S., Wong, A.K. and Wang, Y., 2007. *Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition*, 40(12), pp.3358-3378.

20. Chollet, et al. , Keras (2015).

21. Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. *Tensorflow: A system for large-scale machine learning.* In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283. 2016.

22. Antonakakis, Manos, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. *From throw-away traffic to bots: detecting the rise of DGA-based malware.* In Presented as part of the 21st USENIX Security Symposium (USENIX Security 12), pp. 491-506. 2012.

23. Tran, D., Mac, H., Tong, V., Tran, H.A. and Nguyen, L.G., 2018. *A LSTM based framework for handling multiclass imbalance in DGA botnet detection.* Neurocomputing, 275, pp.2401-2413.

24. Geffner, J., 2013. *End-to-end analysis of a domain generating algorithm malware family.* Black Hat USA, 2013.

25. Choudhary, Chhaya, Raaghavi Sivaguru, Mayana Pereira, Bin Yu, Anderson C. Nascimento, and Martine De Cock. *Algorithmically generated domain detection and malware*

family classification. In International Symposium on Security in Computing and Communication, pp. 640-655. Springer, Singapore, 2018.

26. Rajalakshmi, R., S. Ramraj, and R. Ramesh Kannan. *Transfer Learning Approach for Identification of Malicious Domain Names.*In International Symposium on Security in Computing and Communication, pp. 656-666. Springer, Singapore, 2018.

27. Bharathi, B., and J. Bhuvana. Domain Name Detection and Classification Using Deep Neural Networks.In International Symposium on Security in Computing and Communication, pp. 678-686. Springer, Singapore, 2018.

28. Plohmann, Daniel, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. *A comprehensive measurement study of domain generating malware.* In 25th USENIX Security Symposium (USENIX Security 16), pp. 263-278. 2016.