Chapter 37
DeepDGA-MINet: Cost-Sensitive Deep Learning Based Framework for Handling Multiclass Imbalanced DGA Detection

R. Vinayakumar, K. P. Soman, and Prabaharan Poornachandran

Abstract Contemporary malware families typically use domain generation algorithms (DGAs) to circumvent DNS blacklists, sinkholing, or any types of security system. It means that compromised system generates a large number of pseudo-random domain names by using DGAs based on a seed and uses the subset of domain names to contact the command and control server (C2C). To block the communication point, the security organizations reverse engineer the malware samples based on a seed to identify the corresponding DGA algorithm. Primarily, the lists of reverse engineered domain names are sink-holed and preregistered in a DNS blacklist. This type of task is tedious and moreover DNS blacklist able to detect the already existing DGA based domain name. Additionally, this type of system can be easily circumvented by DGA malware authors. A variant to detect DGA domain name is to intercept DNS packets and identify the nature of domain name based on statistical features. This type of system uses contextual data such as passive DNS and NXDomain. Developing system to detect DGA based on contextual data is difficult due to aggregation of all data and it causes more cost in real-time environment and moreover obtaining the contextual information in end point system is often difficult due to the real-world constraints. Recently, the method which detects the DGA domain name on per domain basis is followed. This method doesn't rely on any external information and uses only full domain name. There are many works for detecting DGA on per domain names based on both manual feature engineering with classical machine learning (CML) algorithms and automatic feature engineering with deep learning architectures. The performance of

R. Vinayakumar (✉) · K. P. Soman
Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: r_vinayakumar@cb.amrita.edu

P. Poornachandran
Centre for Cyber Security Systems and Networks, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
e-mail: praba@amrita.edu

B. B. Gupta et al. (eds.), *Handbook of Computer Networks and Cyber Security*,
https://doi.org/10.1007/978-3-030-22277-2_37

methods based on deep learning architectures is higher when compared to the CML algorithms. Additionally, the deep learning based DGA detection methods can stay safe in an adversarial environment when compared to CML classifiers. However, the deep learning architectures are vulnerable to multiclass imbalance problem. Additionally, the multiclass imbalance problem is becoming much more important in DGA domain detection. This is mainly due to the fact that many DGA families have very less number of samples in the training data set. In this work, we propose DeepDGA-MINet which collects the DNS information inside an Ethernet LAN and uses Cost-Sensitive deep learning architectures to handle multiclass imbalance problem. This is done by initiating cost items into backpropogation methodology to identify the importance among each DGA families. The performances of the Cost-Sensitive deep learning architecture are evaluated on AmritaDGA benchmark data set. The Cost-Sensitive deep learning architectures performed well when compared to the original deep learning architectures.

Keywords Cyber security · Cybercrime · Multiclass imbalance · Cost-Sensitive learning · Deep learning · Domain fluxing · Domain generation algorithm · Malware · Botnet · Malicious domain name

1 Introduction

Contemporary malware families installed on infected computers typically use domain generation algorithms (DGAs) [17]. As stated by DGArchive,[1] there are 72 different DGA families. The DGA families might increase in the near future because it can oppose the botnet takedown mechanism [17]. A malware family which uses DGAs is called as domain fluxing [19]. DGAs support to generate many pseudo-random domain names and a subset of the domain names is used to establish a connection to command and control (C2C) server. To make a successful communication, an author has to register only a small subset of domain names. Based on the successful establishment of communication, the malicious author can start executing malicious activities while the entire information is passed to the botmaster. Then the botmaster issues instructions to bots and sometime even to update the malware family itself. Analysis of DNS traffic provides a way to detect malicious activities hosted by botnet. In recent days, botnet has been used as a primary approach to countermeasure many malicious activities [26].

Recent days, botnet has remained as a serious threat to the Internet service community. Thus detecting DGAs has been a significant problem in the domain of cyber security [11]. DNS blacklisting is the most commonly used method for detecting DGA domain name in earlier days. The significance of DNS blacklisting

[1] https://dgarchive.caad.fkie.fraunhofer.de/.

method for DGA analysis is studied by Kührer et al. [11]. The study used both public and private blacklists. Private blacklisting is prepared by vendors and the experimental results show that the private blacklisting survived better than the public blacklisting. The results of public blacklisting performance varied for DGA malware families. They suggest that the DNS blacklisting is very useful and can be used along with the other approach to provide a more appropriate level of protection. Another approach is to reverse engineer the malware along with its DGA to identify the seed. Once the seed is known, then subsequent domain names can be registered and those registered domain names act as an impersonator C2C server to seize botnets. This type of process is typically called as sinkholing [23]. Once the botnet is seized, an adversary has to redeploy new botnet with revised seeds to further continue the process to do malicious activities. Both blacklisting and sinkholing methods consume more time and resource intensive approaches. More importantly blacklisting completely fails to detect new types of domain name or variants of existing domain name. Sinkholing has low success rate in detecting new types of DGA domain and variants of existing DGA domain name. Later, DGA classifiers are built using machine learning (ML) algorithms. This type of DGA classifier stays in the network and captures the DNS requests and looks for the DGA domain name. Once the DGA domain name is detected, it gives an alert to the network admin to further examine the foundation of a DGA. The existing works on ML based detection are classified into retrospective and real-time. Retrospective detection methods follow clustering and estimate the statistical properties for each cluster for classification [3, 43, 44]. To enhance the system detection rate, retrospective methods use other contextual information WHOIS information, NXDomain, HTTPheaders. Most of the existing methods belong to retrospective detection and contain several issues in deploying in real-time systems [10, 43, 44]. On the other side, real-time detection method acts on domain name only to detect the DGA domain name. Most of the ML based real-time detection methods are based on feature engineering. These methods are easy to evade and require extensive domain knowledge to extract significant features to distinguish the domain name into either legitimate or DGA domain name [20]. In recent days, to avoid feature engineering phase, the application of deep learning is leveraged in the field of cyber security [25, 27–32, 34, 35, 37, 39]. In [42] the authors proposed LSTM based DGA detection and categorization and the method can be deployed in any environment. Generally, the deep learning architectures are prone to multiclass imbalance problems. There are a few DGA families that contain few samples of domain name. Thus the deep learning architectures bias towards the classes which have more number of samples and as a result DGA families which contain very few samples remain undetected. Additionally, deep learning based DGA detection stays safe in an adversarial environment when compared to CML based DGA detection. To handle multiclass imbalance problem, [24] proposed Cost-Sensitive LSTM which performed better than the Cost-Insensitive LSTM architecture. Consequently, in this work we use Cost-Sensitive LSTM and additionally other Cost-Sensitive

deep learning based architecture are considered to evaluate the performances on AmritaDGA[2] data set. The main contributions of the proposed work are given below:

- This work proposes DeepDGA-MINet, which uses Cost-Sensitive deep learning based architectures which can handle the multiclass imbalance in DGA family categorization. The performances of various Cost-Sensitive deep learning based architectures are shown on AmritaDGA data set.
- A detailed experimental analysis of various Cost-Sensitive deep learning based architectures is shown on two different types of testing data sets. These data sets are completely disjoint and include time information. Thus models evaluated on these data sets facilitate to meet zero day malware detection.

The rest of the part of this chapter is organized as follows: Sect. 2 discusses the background details of DNS, botnet, DGA, Keras embedding, deep learning architectures, and Cost-Sensitive deep learning architectures. Section 3 discusses the related works on application of deep learning on DGA analysis. Section 4 discusses the description of data set. Section 5 discusses the statistical measures. Section 6 includes the proposed framework. Section 7 includes experimental results and observations. At last, conclusion, future work, and discussions are presented in Sect. 8.

2 Background

2.1 Domain Name System (DNS)

Domain name system (DNS) is a critical component in an Internet service system. It maintains a distributed database that facilitates to translate domain name to Internet protocol (IP) address and vice versa. Thus DNS is a main component for nearly all network services and has been a main target for attackers. Domain name is a name of a particular application in an Internet service system which follows naming convention system defined by DNS. The maximum length of the domain name is 63 and parts of the domain name are separated by dots. Generally, the right most element in a domain name is root and left most element is the host label. DNS maintains a hierarchy to manage the domain name named as domain name space. The domain name space is divided into different authorities called as DNS zone. The hierarchy is shown in Fig. 37.1 and it represents the organizational structure. The domain name with the host label and root is called as fully qualified domain name (FQDN). Primarily, there are two types of DNS server, they are recursive and non-recursive. Recursive server contacts the nearby DNS server if the requested

[2] https://vinayakumarr.github.io/AmritaDGA/.

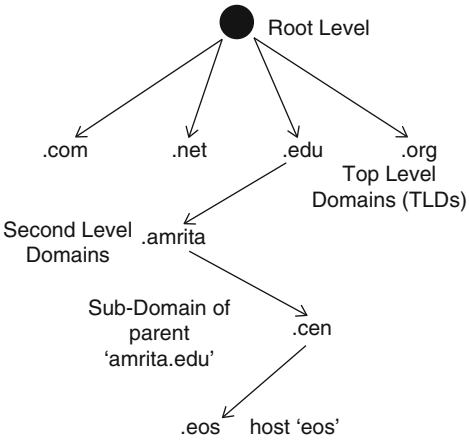

Fig. 37.1 An overview of domain name system

information doesn't exist. Thus there may be a possibility for various attacks such as denial of service (DoS), distributed denial of service (DDoS), DNS cache poisoning, etc.

2.2 Botnet

Botnet is a network of compromised computers that is remotely controlled by botmaster or bot herder. The compromised computers use same malicious code and each compromised computer in a network is called as bot. Botmaster frequently updates the code of bot to evade the current detection methods. A bot uses DGAs to establish a communication channel to a command and control (C2C) server. Recently, botnet behavior is discussed in detail by Alomari et al. [2]. Botnet has been most commonly used by cyber criminals nearly to inject various types of malicious activities and has become a serious threat in the Internet service. Recent botnets use fluxing approach to establish a communication point between bot and C2C server. Mostly, two types of fluxing are used. They are IP flux and domain flux. This work is towards domain flux and domain flux uses the DGA. The DGA algorithm is shared between the botmaster and bots. To establish a connection to C2C server, there may be possibility that DGA generates many failed DNS queries.

Based on the architectures, botnets are grouped into three categories [6]. They are centralized, decentralized, and hybrid. In centralized architecture a botmaster controls all the connected bot in a single point called command and control server (C2C server). Centralized botnet architecture uses star and hierarchical topology and Internet relay chat (IRC) and Hyper Text Transfer Protocol (HTTP) protocols. Decentralized architecture contains more than one C2C server and peer-to-peer protocol. Hybrid architecture is a combination of centralized and decentralized architecture.

2.3 Domain Generation Algorithms (DGAs)

Mostly, recent malware families use DGA instead of hardcoded addresses [17]. This is due to the fact that the DGA is an algorithm which generates large number of pseudo-random domain names based on a seed and appends a top level domain (TLD) such as .com, .edu, etc. to the pseudo-random domain names. A seed can be anything mostly used are data and time information and a seed is shared between the botmaster and bots.

2.4 Domain Name Representation Using Keras Embedding

In this work, Keras embedding is used for domain name representation. In the beginning, a dictionary is formed for the DGA data set which contains only unique characters. Generally, it includes an extra position to handle an unknown character in the testing phase. Each character in a domain name is replaced by a particular index of the dictionary. This transforms the index value in a domain name vector into N dimensional continuous vector representation. The N acts as hyperparameter. This type of representation captures the similarity among the characters in a domain name. The Keras embedding takes the following parameters as input:

- **Dictionary-size:** The number of unique characters
- **Embedding-length:** The size of the embedding vector dimension
- **Input-length:** The size of the input vector

We used Gaussian distribution to initialize the weights during beginning phase in training. The weights are fine-tuned during backpropogation and it coordinatively works with other deep learning layers.

2.5 Deep Learning Architectures

Deep learning is an advanced model of classical machine learning (CML) [13]. They have the capability to obtain optimal feature representation by taking raw input samples. Generally, there are two types of deep learning architectures, one is convolutional neural network (CNN) and another one is recurrent structures (RSs) such as recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU). Primarily CNNs are used on data which includes spatial properties and RSs are used on data which includes time or sequence information. Basic information along with mathematical details for RNN and CNN is discussed below.

Recurrent neural network (RNN), enhanced model of RNN named as long short-term memory (LSTM) [13], minimized version LSTM named as gated recurrent

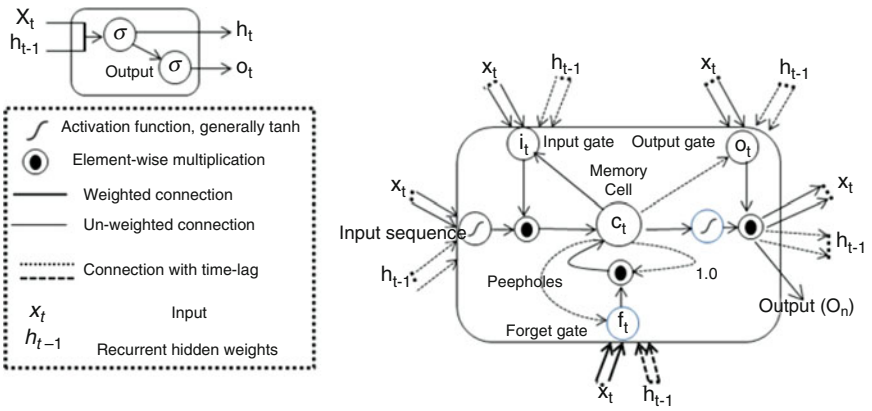

Fig. 37.2 Architecture of recurrent neural network (RNN) unit (left) and Long short-term memory (LSTM) memory block (right)

unit (GRU) [13] belong to the family of RSs. They are most commonly used on sequential data tasks. The structures of RSs look similar to classical feed-forward networks (FFN) and additionally the neurons in RSs contain a self-recurrent connection. All RSs are trained using backpropagation through time (BPTT). RNN in RSs generates vanishing and exploding gradient issue when the network is trained for longer time-steps [13]. To handle vanishing and exploding gradient issue, LSTM was introduced. It replaces the simple RNN unit with a memory block. This has the capability to carry out the important information across time-steps. A memory unit contains a memory cell and gating functions such as input gate, output gate, and forget gate. All 3 different gating functions control a memory cell. However, LSTM contains more parameters. Later a minimized version of LSTM, named as GRU is introduced. GRU achieves the same performance as LSTM and additionally it is computationally inexpensive. A basic unit in RNN, LSTM, and GRU is shown in Figs. 37.2 and 37.3, respectively. The computational functions for RNN, LSTM, and GRU are defined mathematically as follows:

Generally RSs take input $x = (x_1, x_2, \ldots, x_T)$ (where $x_t \in R^d$) and maps to hidden input sequence $h = (h_1, h_2, \ldots, h_T)$ and output sequences $o = (o_1, o_2, \ldots, o_T)$ from $t = 1$ to T by iterating the following equations:

Recurrent Neural Network (RNN)

$$h_t = \sigma(w_{xh}x_t + w_{hh}h_{t-1} + b_h) \tag{37.1}$$

$$o_t = sf(w_{ho}h_t + b_o) \tag{37.2}$$

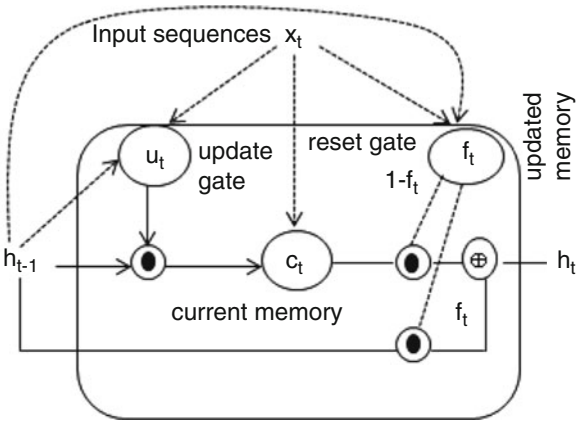

Fig. 37.3 Architecture of unit in gated recurrent unit (GRU)

Long Short-Term Memory (LSTM)

$$i_t = \sigma(w_{xi}x_t + w_{hi}h_{t-1} + w_{ci}c_{t-1} + b_i) \tag{37.3}$$

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + w_{cf}c_{t-1} + b_f) \tag{37.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \tag{37.5}$$

$$o_t = \sigma(w_{xo}x_t + w_{ho}h_{t-1} + w_{co}c_t + b_o) \tag{37.6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{37.7}$$

Gated Recurrent Unit (GRU)

$$u_t = \sigma(w_{xu}x_t + w_{hu}h_{t-1} + b_u) \tag{37.8}$$

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \tag{37.9}$$

$$c_t = \tanh(w_{xc}x_t + w_{hc}(f \odot h_{t-1}) + b_c) \tag{37.10}$$

$$h_t = f \odot h_{t-1} + (1 - f) \odot c \tag{37.11}$$

where w terms for weight matrices, b terms for bias, σ is the $sigmoid$ activation function, sf at output layer denotes the $softmax$ activation function, $tanh$ denotes

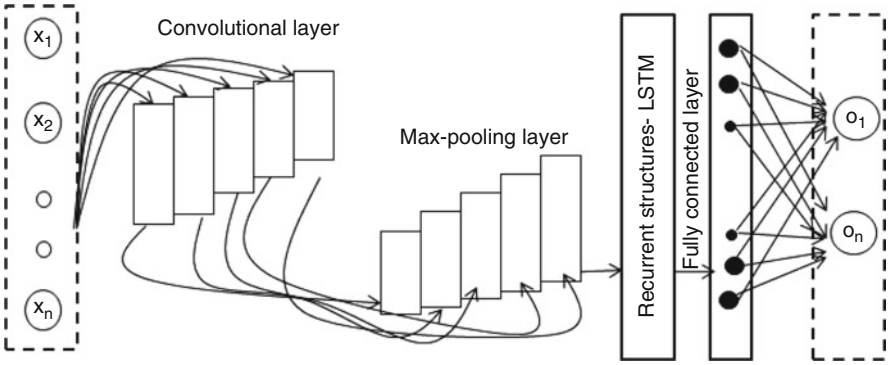

Fig. 37.4 An overview of combination of convolutional neural network (CNN) and long short-term memory (LSTM) architectures

the $tanh$ activation function, i, h, f, o, c denotes the input, hidden, forget, output, and cell activation vectors, in GRU input gate and forget gate are combined and named as update gate u.

Convolutional neural network (CNN) is a type of deep learning architecture which is most commonly used in spatial data analysis [13]. Primarily, CNN is composed of three different sections, they are convolution, pooling, and fully connected layer. Convolution operation is composed of convolution and filters that slide over the domain name vector and extracts the features. The collection of features of convolutional layer is called as feature map. The feature map is huge and to reduce the dimension pooling is used, pooling can be max, min, or average pooling. Finally, the reduced feature representation is passed into fully connected layer for classification. Moreover, the pooling layer can also be passed into RSs to extract the sequence information among the character in the domain name. This type of hybrid architecture is shown in Fig. 37.4.

2.6 Employing Cost-Sensitive Model for Deep Learning Architectures to Handle Multiclass Imbalance Problem

All deep learning architectures focus on minimizing the cost function of the network by considering the true output y^l and a target t^l, where l defines the number of neurons and let's define the cost function for $softmax$.

$$E(t) = -\sum_{S \in samples} \sum_{l} t^l(t) \log y^l(t) \tag{37.12}$$

Generally gradient descent with truncated version of real-time recurrent learning (RTRL) is used to minimize the cost function [13]. As Eq. (37.12) indicates that

the deep learning architectures consider all the samples of each class equally. Thus, deep learning architectures are more prone to class imbalance problem. This type of architectures biased towards the classes which has more number of samples and shows less performance for detecting DGA families which contains less representation in training data set [42]. Cost-Sensitive learning is an important approach in many of the real-time data mining applications and capable to handle multiclass imbalance problem [18].

There are various methods exist to convert the Cost-Insensitive LSTM to Cost-Sensitive method [48]. One of the most commonly used methods is to accommodate the balanced training samples via following oversampling or under sampling [49]. In [48] the authors reported the resampling approach is not an efficient method in dealing with class imbalance on multiclass applications. Later, several methods were introduced based on threshold. In [12] the authors proposed Cost-Sensitive based neural networks to handle multiclass imbalance problem by using the error minimization function with the aim to achieve the expected costs. They haven't mainly targeted the class imbalance problem in their experiment. Following [24], introduced Cost-Sensitive LSTM which incorporates the misclassification costs into the backward pass of LSTM. Each sample S is coupled with a cost item $c[class(S), k]$, where k and $class(S)$ define the predicted and actual class, respectively. A cost weight is assigned based on the frequency of samples of a class. Generally, the cost items indicate the classification importance.

$$E(t) = -\sum_{S \in samples} \sum_{l} t^l(t) \log y^l(t) c[class(S), k \tag{37.13}$$

Based on Eq. (37.13), the basic equations for all deep learning architectures are changed by including the cost item. A cost item typically controls the magnitude of weight updates [9].

Initially for an input data samples the cost matrix is not known. Application of genetic algorithm can be used to identify the optimal cost matrix. However, it requires more time and considered as a difficult task [41]. Let's assume the data samples in one type of class are equal cost. $C[i, i]$ indicates the misclassification cost of the class i, which is produced using the class distribution as

$$c[i, i] = \left(\frac{1}{n_i}\right)^{\gamma} \tag{37.14}$$

where $\gamma \in [0, 1]$ is a hyperparameter, if $c[i, i]$ is inversely proportional to the class size n_i, then $\gamma = 1$ amd $\gamma = 0$ indicate the deep learning architectures are Cost-Insensitive.

3 Related Works on Domain Generation Algorithms (DGAs) Analysis

A detailed review of detecting malicious domain names is reported by Zhauniarovich et al. [47]. In earlier days, blacklisting is the most commonly used method. These methods completely fail to detect new kinds or variants of DGA based domain name. Later, many approaches have been introduced based on machine learning (ML). These ML based solutions are mostly retrospective which means the methods build clusters based on the statistical properties [3, 43, 44]. These methods are not efficient in real-time DGA domain name detection. Additionally, the retrospective methods take advantage of additional information obtained from HTTP headers, NXDomains, and passive DNS information. Later, real-time detection based on ML is introduced. These methods act on a per domain information which means extract different features from domain name and pass into ML algorithms for classification [20]. However, these ML based solutions rely on feature engineering. This is considered as one of the daunting tasks and these solutions are vulnerable in an adversarial environment.

Recently the application of deep learning is leveraged for DGA detection which completely avoids feature engineering [33, 42]. In [42] the authors proposed a method for DGA detection and categorization. The method uses LSTM which looks for DGA domain name on per domain bases. The method performed well when compared to the benchmark classical methods based on HMM and also results are compared with the feature engineering methods. In [33] the authors proposed a method to collect DNS logs inside an Ethernet LAN and to analyze the DNS logs the application of deep learning architectures such as RNN and LSTM was used. The results are compared with the classical method, feature engineering with Random Forest classifiers. A detailed experimental analysis is shown for various data sets collected in real-time and public sources. The application of various deep learning architectures such as RNN, GRU, LSTM, CNN, and CNN-LSTM is evaluated for DGA detection and categorization [40]. For comparative study bigram with logistic regression and feature engineering with Random Forest classifier is mapped. In all the experiments, the deep learning architectures performed well when compared to the classical methods. In [26] the authors developed a cyber-threat situational awareness framework by using DNS data. They showed a method to collect the DNS logs at an Internet service provider level and application of deep learning architecture is used for DNS data analysis with the aim to detect the DGA domain names. In [45] the authors proposed a method to automatically label the data into DGA or non-DGA and used deep learning architecture for DNS data analysis. For comparative study, 11 different feature sets are extracted based on the domain knowledge and passed into Random Forest classifier. A detailed study of all the different models was evaluated on very large volume of data set which was collected from both the public source and real-time DNS streams. The deep learning model particularly CNN performed well when compared to feature based approach and the system performance has been shown on live stream deployment. In [14] the authors evaluated the performance of recurrent networks on very large

volume of data set which consists of 61 different DGA malware families. In recent days, many deep learning architectures based on character level embedding are introduced for many text applications in the field of NLP. To leverage the application of these models [46] evaluated the performance of various benchmark deep learning architectures with character based models for DGA detection and compared with classical methods, feature engineering with Random Forest and multilayer perceptron (MLP) classifiers. The methods based on deep learning with character level embedding performed better than the classical methods. The application of various Image Net models such as AlexNet, VGG, SqueezeNet, InceptionNet, ResNet are transformed for DGA detection by Feng et al. [7]. They followed preprocessing approach to convert the domain name into image format and followed transfer learning approach. In [15] the authors evaluated the performance of various supervised learning models such as LSTM, recurrent SVM, CNN with LSTM, and bidirectional LSTM and compared it with the classical methods HMM, C4.5, ELM, and SVM on the 38 DGA families data set which was collected in real-time. In [5] the authors proposed a method which uses recurrent networks for DGA detection. The method takes the benefit of side information from WHOIS database. This is due to the fact that the DGA families with a high average Smashword score are very difficult to detect based on the domain information alone in the case of a per domain basis method. Smashword score defines the average of n-gram (n ranges [3–5]) intersection with words from an English word dictionary. Generally, it is the measure that gives the measure of closeness between DGA and English words. In [24] the authors proposed Cost-Sensitive LSTM to handle multiclass imbalance in DGA families detection. The proposed method showed 7% improvement in both precision and recall when compared to the Cost-Insensitive LSTM. Additionally, the Cost-Sensitive LSTM showed better performance in detecting 5 additional DGA based bot families. In [22] the authors evaluated the performance of various benchmark character based models for DGA detection and categorization. These models are based on ensemble of human engineered and machine learned features. The importance to time and seed is given while selecting the data set for train and test. Thus this type of methodology allows effectively to evaluate the robustness of the trained classifiers for identifying domain names initiated by the same families at various times or even seed changes. They also state that their method performed well for detecting DGA in the case of time dependent seed when compared to time invariant DGAs. They also evaluated the best performed model on real-time DNS traffic and showed that many of the legitimate domain names are flagged as legitimate. This is mainly due to the reason that Alexa is not completely a non-malicious domain name in real-time DNS traffic. In [16] the authors proposed a unique deep learning architecture typically called as spoofnet which correlates both DNS and URL data to detect malicious activities. Following, the spoofnet architecture is evaluated on various types of data sets of DGA and URL and additionally employed for spam email detection [38]. To meet zero day malware detection, [37] incorporated the time information in generating the data sets for train and test. To leverage the application of various character based benchmark models, [35] transformed these approaches to DGA analysis.

4 Description of Data Set

To measure the performance of Cost-Sensitive based deep learning architectures, we have used the AmritaDGA[3] data set [36]. This data set has been used as part of DMD-2018 shared task. Along with the data set, baseline system[4] is publically available for further research. This data set contains domain names which are collected from publically available sources and real-time DNS traffic inside an Ethernet LAN. Additionally, the data set has been designed by giving importance to the time information. Thus, the trained models on this type of data set have the ability to meet zero day malware detection. The data set is composed of two types of testing data sets. Testing 1 data set is formed using publically available sources and Testing 2 data set is formed using DNS traffic inside an Ethernet LAN. The statistics of AmritaDGA is shown in Table 37.1. The data set was used for two tasks, one is binary and other is multiclass classification. Binary class classification

Table 37.1 AmritaDGA data set used in DMD-2018 shared task

Class	Training	Testing 1	Testing 2
benign	100,000	120,000	40,000
banjori	15,000	25,000	10,000
corebot	15,000	25,000	10,000
dircrypt	15,000	25,000	300
dnschanger	15,000	25,000	10,000
fobber	15,000	25,000	800
murofet	15,000	16,667	5000
necurs	12,777	20,445	6200
newgoz	15,000	20,000	3000
padcrypt	15,000	20,000	3000
proslikefan	15,000	20,000	3000
pykspa	15,000	25,000	2000
qadars	15,000	25,000	2300
qakbot	15,000	25,000	1000
ramdo	15,000	25,000	800
ranbyus	15,000	25,000	500
simda	15,000	25,000	3000
suppobox	15,000	20,000	1000
symmi	15,000	25,000	500
tempedreve	15,000	25,000	100
tinba	15,000	25,000	700
Total	397,777	587,112	103,200

[3] https://vinayakumarr.github.io/AmritaDGA/.

[4] https://github.com/vinayakumarr/DMD2018.

aims at classifying the domain name as either legitimate or DGA and multiclass categorizes the domain name to their families.

5 Statistical Measures

To measure the performance of trained models of various deep learning architectures, we adopted the various statistical measures. These various measures are approximated based on the positive (PD) : legitimate domain name, negative (NG): DGA domain name, true positive (T_{PD}) : legitimate domain name that is predicted as legitimate, true negative (T_{ND}) : DGA domain name that is predicted as DGA domain name, false positive (F_{PD}) : DGA domain name that is predicted as legitimate, and false negative (F_{ND}) : legitimate domain name that is predicted as DGA domain name. Using confusion matrix T_{PD}, T_{ND}, F_{PD}, and F_{ND} are obtained. Confusion matrix is represented in the form of matrix where each row denotes the domain name samples of a predicted class and each column denotes domain name samples of actual class. The various statistical measures considered in this study are defined as follows:

$$Accuracy = \frac{T_{PD} + T_{ND}}{T_{PD} + T_{ND} + F_{PD} + F_{ND}} \tag{37.15}$$

$$Recall = \frac{T_{PD}}{T_{PD} + F_{ND}} \tag{37.16}$$

$$Precision = \frac{T_{PD}}{T_{PD} + F_{PD}} \tag{37.17}$$

$$F1{-}score = \frac{2 * \text{Re}call * \text{Pr}ecision}{\text{Re}call + \text{Pr}ecision} \tag{37.18}$$

$$TPR = \frac{T_{PD}}{T_{PD} + F_{PD}} \tag{37.19}$$

$$FPR = \frac{F_{PD}}{F_{PD} + T_{ND}} \tag{37.20}$$

$Accuracy$ estimates the fraction of correctly classified domain name, $Precision$ estimates the fraction of DGA domain name which is actually DGA domain name, $Recall$ or $Sensitivity$ or TPR estimates the fraction of DGA domain names that are classified as DGA domain name, and $F1$-score estimates the harmonic mean of precision and recall.

6 Proposed Architecture: DeepDGA-MINet

The proposed architecture named as DeepDGA-MINet is shown in Fig. 37.5. A detailed overview of DeepDGA-MINet is shown in Fig. 37.6. This contains mainly 3 different sections: (1) data collection, (2) Cost-Sensitive deep learning layers, and (3) classification.

In data collection, the system collects the DNS logs inside an Ethernet LAN in a passive way. The data has been passed into NoSQL database. Further, the domain name information is extracted from the DNS logs and passed into the Cost-Sensitive deep learning layers. This implicitly composed of character level embedding layer which helps to map the domain name characters into domain name numeric

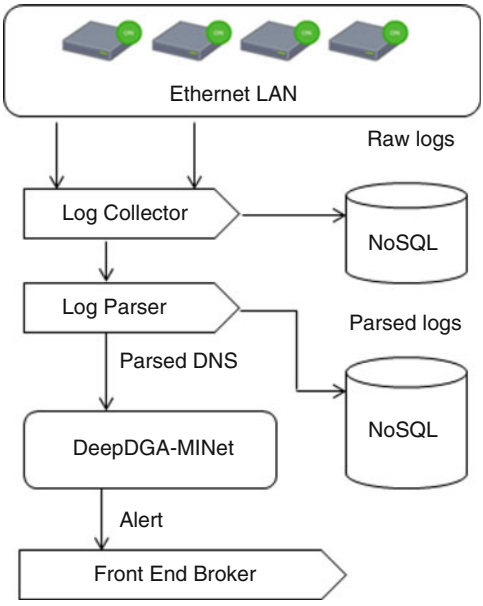

Fig. 37.5 Proposed architecture: DeepDGA-MINet

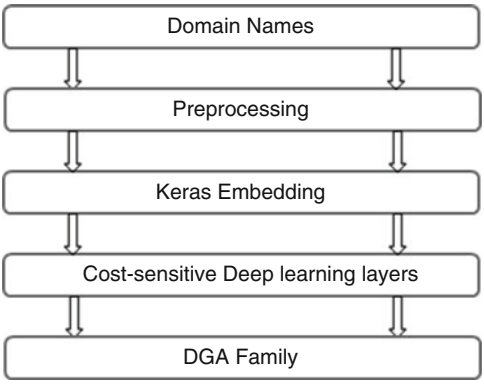

Fig. 37.6 A detailed overview of DeepDGA-MINet

representation. The character level embedding layer works with Cost-Sensitive deep learning layers to extract the similarity among characters during backpropogation. Further, Cost-Sensitive deep learning layer extracts significant features from the character level embedding vectors. Finally, the feature set is passed into the fully connected layer for classification. This composed of $softmax$ activation function which uses categorical cross-entropy loss function. The $softmax$ and categorical cross-entropy loss function are defined mathematically as follows:

$$Soft\max(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{37.21}$$

$$loss(p, e) = -\sum_x p(x)\log(e(x)) \tag{37.22}$$

where x denotes an input , e and p denote true probability distribution and predicted probability distribution, respectively. To minimize the loss function $adam$ optimizer is used. Finally, the classification results are displayed in Front End Broker.

7 Experiments, Results, and Observations

The detailed configuration details of deep learning architectures are reported in Table 37.2. In this research all the deep learning architectures are implemented in TensorFlow [1] with Keras [4] higher level library and various experiments of deep learning architectures are run on GPU enabled TensorFlow inside Nvidia GK110BGLTeslak40. All deep learning architectures are trained using AmritaDGA data set. To control the train accuracy across the more number of epochs, we have used validation data set that was from 20% of train data set taken randomly. The domain name samples are transformed into numeric vectors using Keras embedding. Keras embedding implicitly builds a dictionary which contains 39 unique characters. The character list is given below:

abcdefghijklmnopqrstuvwxyz0123456789._ -

Using dictionary the characters of a domain name are transformed into indexes. The maximum length of the domain name is 91. Thus the domain name which contains less than 91 is padded with 0s. The index vector is passed into Keras embedding. It takes 3 different parameters such as Dictionary-size is 39, Embedding-length is 128, and Input-length is 91. Keras embedding follows deep learning layers and the detailed configuration details of deep learning layers are reported in Table 37.2. The deep learning layers follow fully connected layer for classification. All the trained models of various deep learning architectures are tested on the two types of AmritaDGA data set and the detailed results are reported in Tables 37.3, 37.4, 37.5, and 37.6. All DMD-2018 shared tasks submitted systems have used Cost-Insensitive deep learning architectures. The proposed deep learning architectures based on

Table 37.2 Detailed configuration of deep learning architectures

Layer (type)	Output shape	Param #
RNN		
embedding_1 (Embedding)	(None, 91, 128)	5120
simple_rnn_1 (SimpleRNN)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 21)	2709
activation_1 (Activation)	(None, 21)	0
Total params: 40,725		
LSTM		
embedding_1 (Embedding)	(None, 91, 128)	5120
lstm_1 (LSTM)	(None, 128)	131,584
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 21)	2709
activation_1 (Activation)	(None, 21)	0
Total params: 139,413		
GRU		
embedding_1 (Embedding)	(None, 91, 128)	5120
gru_1 (GRU)	(None, 128)	98,688
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 21)	2709
activation_1 (Activation)	(None, 21)	0
Total params: 106,517		
CNN		
embedding_1 (Embedding)	(None, 91, 128)	5120
conv1d_1 (Conv1D)	(None, 87, 64)	41,024
max_pooling1d_1 (MaxPooling1)	(None, 21, 64)	0
dense_1 (Dense)	(None, 21, 128)	8320
dropout_1 (Dropout)	(None, 21, 128)	0
activation_1 (Activation)	(None, 21, 128)	0
dense_2 (Dense)	(None, 21, 21)	2709
activation_2 (Activation)	(None, 21, 21)	0
Total params: 57,173		
CNN-LSTM		
embedding_1 (Embedding)	(None, 91, 128)	5120
conv1d_1 (Conv1D)	(None, 87, 64)	41,024
max_pooling1d_1 (MaxPooling1)	(None, 21, 64)	0
lstm_1 (LSTM)	(None, 70)	37800
dense_1 (Dense)	(None, 21)	1491
activation_1 (Activation)	(None, 21)	0
Total params: 85,435		

Table 37.3 Detailed Testing 1 data set results of DMD-2018 shared task participated systems [36]

Team name	Accuracy (%)	Recall (%)	Precision (%)	$F1$-score (%)
UWT	63.3	63.3	61.8	60.2
Deep_Dragons	68.3	68.3	68.3	64
CHNMLRG	64.8	64.8	66.2	60
BENHA	27.2	27.2	19.4	16.8
BharathibSSNCSE	18	18	9.2	10.2
UniPI	65.5	65.5	64.7	61.5
Josan	69.7	69.7	68.9	65.8
DeepDGANet	60.1	60.1	62	57.6

Table 37.4 Detailed Testing 2 data set results of DMD-2018 shared task participated systems [36]

Team name	Accuracy (%)	Recall (%)	Precision (%)	$F1$-score (%)
UWT	88.7	88.7	92.4	90.1
Deep_Dragons	67	67	67.8	62.2
CHNMLRG	67.4	67.4	68.3	64.8
BENHA	42.9	42.9	34	27.2
BharathibSSNCSE	33.5	33.5	22.9	22.3
UniPI	67.1	67.1	64.1	61.9
Josan	67.9	67.9	69.4	63.6
DeepDGANet	53.1	53.1	65.3	54.1

Table 37.5 Detailed Testing 1 data set results of AmritaDGA baseline system for multiclass classification [36]

Architecture	Accuracy (%)	Precision (%)	Recall (%)	$F1$-score (%)
RNN	65.8	63.6	65.8	62.6
LSTM	67.2	66.3	67.2	62.2
GRU	64.9	65.5	64.9	60.1
CNN	60.4	62.9	60.4	56.8
CNN-LSTM	59.9	61.5	59.9	55.6

Table 37.6 Detailed Testing 2 data set results of AmritaDGA baseline system for multiclass classification [36]

Architecture	Accuracy (%)	Precision (%)	Recall (%)	$F1$-score (%)
RNN	66.2	62.7	66.2	60.9
LSTM	66.9	69.5	66.9	62.7
GRU	66.5	71.8	66.5	63.7
CNN	64.3	69.1	64.3	59.6
CNN-LSTM	65.8	67.6	65.8	62.5

Table 37.7 Detailed Testing 1 data set results of the proposed method—deep learning architectures based on Cost-Sensitive data mining concept

Architecture	Accuracy (%)	Precision (%)	Recall (%)	$F1$-score (%)
Cost-Sensitive-RNN	65.8	63.6	65.8	62.6
Cost-Sensitive-LSTM	68.3	65.8	68.3	64.0
Cost-Sensitive-GRU	68.3	65.8	68.3	66.0
Cost-Sensitive-CNN	62.8	63.8	62.8	59.5
Cost-Sensitive-CNN-LSTM	64	64.3	64	62

Table 37.8 Detailed Testing 2 data set results of the proposed method—deep learning architectures based on Cost-Sensitive data mining concept

Architecture	Accuracy (%)	Precision (%)	Recall (%)	$F1$-score (%)
Cost-Sensitive-RNN	65.8	63.6	65.8	62.6
Cost-Sensitive-LSTM	67.3	69.9	67.3	63.1
Cost-Sensitive-GRU	67.6	67.6	67.6	63.9
Cost-Sensitive-CNN	67.1	74.8	67.1	65.6
Cost-Sensitive-CNN-LSTM	67.1	70.7	67.1	64.7

Cost-Sensitive performed better than the baseline system of DMD-2018 and all the submitted entries of DMD-2018 shared task, as shown in Tables 37.7 and 37.8. The detailed results for Testing 1 AmritaDGA data set are reported in Tables 37.9 and 37.10 for Testing 2 AmritaDGA data set. All baseline system of DMD-2018 and all the submitted entries of DMD-2018 shared task methods are based on Cost-Insensitive models. The Cost-Sensitive models can even perform well in detecting real-time DGA. This is due to the reason that most of the data set in real-time are highly imbalanced. This work has given importance only to achieve the best performance when compared to the baseline system and other submitted system entries of DMD-2018 shared task. However, the proposed method can perform well in any other data set and real-time detection of DGA domain name. Mostly, the results obtained by all the models are closer in nature. Moreover, the LSTM model has outperformed other deep learning architectures. However, the reported results can be further enhanced by following parameter tuning method. This is due to the reason that the optimal parameters implicitly have direct impact on getting the best performance in deep learning [13].

8 Conclusion, Future Works, and Discussions

This work proposes DeepDGA-MINet tool which provides an option to collect a live stream of DNS queries and checks for DGA domain name on a per domain basis. It uses the application of Cost-Sensitive deep learning based methods to handle multiclass imbalance problem. Each class or DGA family is associated with cost items and these are directly initiated into backpropogation learning algorithm. The

Table 37.9 Class-wise test results of the proposed method for Testing 1 data set of AmritaDGA

	Cost-Sensitive RNN		Cost-Sensitive LSTM		Cost-Sensitive GRU		Cost-Sensitive CNN		Cost-Sensitive CNN-LSTM	
Classes	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
benign	0.056	0.906	0.069	0.947	0.016	0.864	0.057	0.937	0.048	0.855
banjori	0.004	0.0	0.001	0.0	0.001	0.0	0.0	0.0	0.001	0.0
corebot	0.001	0.996	0.004	1.0	0.008	0.999	0.002	0.998	0.004	0.999
dircrypt	0.036	0.712	0.029	0.817	0.045	0.767	0.035	0.594	0.066	0.631
dnschanger	0.052	0.988	0.051	0.994	0.053	0.993	0.082	0.863	0.061	0.993
fobber	0.009	0.0	0.008	0.0	0.008	0.0	0.024	0.0	0.02	0.0
murofet	0.075	0.0	0.061	0.0	0.067	0.0	0.003	0.006	0.009	0.0
necurs	0.004	0.839	0.004	0.86	0.009	0.857	0.021	0.762	0.015	0.644
newgoz	0.001	0.99	0.001	0.999	0.0	1.0	0.017	1.0	0.002	1.0
padcrypt	0.001	0.99	0.0	1.0	0.0	1.0	0.0	0.999	0.001	1.0
proslikefan	0.018	0.673	0.014	0.689	0.022	0.71	0.013	0.633	0.022	0.506
pykspa	0.034	0.738	0.033	0.886	0.034	0.771	0.031	0.663	0.031	0.712
qadars	0.001	0.764	0.0	0.119	0.001	0.892	0.0	0.302	0.0	0.528
qakbot	0.036	0.426	0.043	0.605	0.021	0.372	0.058	0.486	0.061	0.309
ramdo	0.0	0.998	0.0	1.0	0.0	1.0	0.001	0.999	0.0	1.0
ranbyus	0.004	0.854	0.003	0.874	0.003	0.842	0.008	0.711	0.002	0.75
simda	0.001	0.001	0.0	0.001	0.0	0.0	0.0	0.35	0.002	0.309
suppobox	0.005	0.742	0.002	0.812	0.005	0.95	0.002	0.823	0.008	0.612
symmi	0.0	0.176	0.0	0.601	0.0	0.613	0.0	0.152	0.0	0.585
tempedreve	0.018	0.124	0.015	0.135	0.035	0.131	0.023	0.178	0.025	0.114
tinba	0.008	0.922	0.003	0.97	0.003	0.886	0.017	0.573	0.005	0.966
Accuracy (%)	65.8		68.3		68.3		62.8		64.0	

proportion of cost is a hyperparameter and selected based on hyperparameter tuning method. The performance obtained by Cost-Sensitive based deep learning architectures is good when compared to the Cost-Insensitive deep learning architectures. Moreover, the performance shown by various Cost-Sensitive deep learning based architectures is almost similar. Hence, a voting methodology can be employed to enhance the DGA domain detection rate. This remains as one of the significant direction towards future work. This work has considered only 20 DGA families. The performance is shown for classifying a domain name into corresponding DGA family. Therefore, the further research on investigating the performance of Cost-Sensitive deep learning architectures on more number of DGA families remain as a significant direction towards future work. As well as in this work the hyperparameter tuning is not followed for deep learning architectures. Hyperparameters have direct impact on the performance of deep learning architectures. Thus investigation of proper hyperparameter tuning remains as another significant direction towards future work.

Table 37.10 Class-wise test results of the proposed method for Testing 2 data set of AmritaDGA

	Cost-Sensitive RNN		Cost-Sensitive LSTM		Cost-Sensitive GRU		Cost-Sensitive CNN		Cost-Sensitive CNN-LSTM	
Classes	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
benign	0.192	0.956	0.106	0.984	0.106	0.978	0.088	0.944	0.098	0.979
banjori	0.001	0.0	0.0	0.0	0.0	0.0	0.0	0.336	0.0	0.0
corebot	0.0	0.229	0.0	0.228	0.0	0.228	0.0	0.227	0.0	0.228
dircrypt	0.058	0.7	0.101	0.797	0.092	0.767	0.055	0.45	0.105	0.41
dnschanger	0.011	0.988	0.01	0.994	0.01	0.99	0.022	0.901	0.012	0.987
fobber	0.001	0.0	0.001	0.0	0.002	0.0	0.006	0.0	0.003	0.0
murofet	0.003	0.0	0.001	0.0	0.002	0.0	0.001	0.001	0.001	0.0
necurs	0.029	0.838	0.023	0.857	0.024	0.854	0.015	0.662	0.011	0.619
newgoz	0.008	0.99	0.05	1.0	0.055	1.0	0.06	0.999	0.064	0.999
padcrypt	0.017	0.99	0.0	1.0	0.0	1.0	0.001	0.997	0.0	0.999
proslikefan	0.007	0.332	0.004	0.33	0.005	0.4	0.019	0.542	0.012	0.679
pykspa	0.033	0.735	0.031	0.89	0.029	0.882	0.038	0.78	0.022	0.65
qadars	0.001	0.497	0.0	0.049	0.0	0.183	0.002	0.117	0.0	0.391
qakbot	0.032	0.399	0.029	0.601	0.025	0.516	0.016	0.328	0.01	0.376
ramdo	0.0	0.996	0.0	1.0	0.0	1.0	0.0	0.968	0.0	1.0
ranbyus	0.001	0.848	0.001	0.87	0.001	0.872	0.008	0.684	0.003	0.756
simda	0.001	0.0	0.0	0.017	0.0	0.0	0.003	0.001	0.0	0.224
suppobox	0.003	0.787	0.001	0.89	0.002	0.918	0.005	0.935	0.003	0.521
symmi	0.0	0.956	0.0	0.974	0.0	0.998	0.0	0.994	0.0	0.968
tempedreve	0.013	0.17	0.014	0.16	0.017	0.17	0.016	0.1	0.024	0.17
tinba	0.004	0.129	0.001	0.283	0.001	0.279	0.012	0.386	0.004	0.657
Accuracy (%)	65.8		67.3		67.6		67.1		67.1	

Acknowledgements This research was supported in part by Paramount Computer Systems and Lakhshya Cyber Security Labs. We are grateful to NVIDIA India, for the GPU hardware support to research grant. We are also grateful to Computational Engineering and Networking (CEN) department for encouraging the research.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). TensorFlow: A system for large-scale machine learning. In *OSDI* (Vol. 16, pp. 265–283).
2. Alomari, E., Manickam, S., Gupta, B. B., Anbar, M., Saad, R. M., & Alsaleem, S. (2016). A survey of botnet-based DDoS flooding attacks of application layer: Detection and mitigation approaches. In *Handbook of research on modern cryptographic solutions for computer and cyber security* (pp. 52–79). Pennsylvania, PA: IGI Global.
3. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., et al. (2012). From throw-away traffic to bots: Detecting the rise of DGA-based malware. In *USENIX Security Symposium* (Vol. 12).
4. Gulli, A., & Pal, S. (2017). Deep Learning with Keras. Packt Publishing Ltd.

5. Curtin, R. R., Gardner, A. B., Grzonkowski, S., Kleymenov, A., & Mosquera, A. (2018). Detecting DGA domains with recurrent neural networks and side information. arXiv preprint arXiv:1810.02023.
6. Eslahi, M., Salleh, R., & Anuar, N. B. (2012). Bots and botnets: An overview of characteristics, detection and challenges. In *2012 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)* (pp. 349–354). Piscataway, NJ: IEEE.
7. Feng, Z., Shuo, C., & Xiaochuan, W. (2017). Classification for DGA-based malicious domain names with deep learning architectures. In *2017 Second International Conference on Applied Mathematics and Information Technology* (p. 5).
8. Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *ICML* (Vol. 96, pp. 148–156).
9. He, H., & Garcia, E. A. (2008). Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering, 21*(9), 1263–1284.
10. Krishnan, S., Taylor, T., Monrose, F., & McHugh, J. (2013). Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing. In *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 1–12). Piscataway, NJ: IEEE.
11. Kührer, M., Rossow, C., & Holz, T. (2014). Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 1–21). Cham: Springer.
12. Kukar, M., & Kononenko, I. (1998). Cost-sensitive learning with neural networks. In *ECAI* (pp. 445–449).
13. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436.
14. Lison, P., & Mavroeidis, V. (2017). Automatic detection of malware-generated domains with recurrent neural models. arXiv preprint arXiv:1709.07102.
15. Mac, H., Tran, D., Tong, V., Nguyen, L. G., & Tran, H. A. (2017). DGA botnet detection using supervised learning methods. In *Proceedings of the Eighth International Symposium on Information and Communication Technology* (pp. 211–218). New York, NY: ACM.
16. Mohan, V. S., Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Spoof net: Syntactic patterns for identification of ominous online factors. In *2018 IEEE Security and Privacy Workshops (SPW)* (pp. 258–263). Piscataway, NJ: IEEE.
17. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., & Gerhards-Padilla, E. (2016). A comprehensive measurement study of domain generating malware. In *USENIX Security Symposium* (pp. 263–278).
18. Qiu, C., Jiang, L., & Kong, G. (2015). A differential evolution-based method for class-imbalanced cost-sensitive learning. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Piscataway, NJ: IEEE.
19. Schiavoni, S., Maggi, F., Cavallaro, L., & Zanero, S. (2014). Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 192–211). Cham: Springer.
20. Schüppen, S., Teubert, D., Herrmann, P., & Meyer, U. (2018). FANCI: Feature-based automated NXDomain classification and intelligence. In *27th USENIX Security Symposium (USENIX Security 18)* (pp. 1165–1181).
21. Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 40*(1), 185–197.
22. Sivaguru, R., Choudhary, C., Yu, B., Tymchenko, V., Nascimento, A., & De Cock, M. (2018). An evaluation of DGA classifiers. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 5058–5067). Piscataway, NJ: IEEE.
23. Stone-Gross, B., Cova, M., Gilbert, B., Kemmerer, R., Kruegel, C., & Vigna, G. (2011). Analysis of a botnet takeover. *IEEE Security & Privacy, 9*(1), 64–72.
24. Tran, D., Mac, H., Tong, V., Tran, H. A., & Nguyen, L. G. (2018). A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing, 275*, 2401–2413.

25. Vinayakumar, R., Barathi Ganesh, H. B., & Anand Kumar, M., Soman, K. P. DeepAnti-PhishNet: Applying deep neural networks for phishing email detection cen-aisecurity@iwspa-2018 (pp. 40–50). http://ceur-ws.org/Vol2124/#paper_9
26. Vinayakumar, R., Poornachandran, P., & Soman, K. P. (2018). Scalable framework for cyber threat situational awareness based on domain name systems data analysis. In *Big Data in Engineering Applications* (pp. 113–142). Singapore: Springer.
27. Vinayakumar, R., & Soman, K. P. (2018). DeepMalNet: Evaluating shallow and deep networks for static PE malware detection. *ICT Express, 4(4)*, 255–258.
28. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1222–1228). Piscataway, NJ: IEEE.
29. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Evaluating effectiveness of shallow and deep networks to intrusion detection system. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 1282–1289). Piscataway, NJ: IEEE.
30. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Evaluation of recurrent neural network and its variants for intrusion detection system (IDS). *International Journal of Information System Modeling and Design, 8*(3), 43–63.
31. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Long short-term memory based operation log anomaly detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 236–242). Piscataway, NJ: IEEE.
32. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Secure shell (SSH) traffic analysis with flow based features using shallow and deep networks. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 2026–2032). Piscataway, NJ: IEEE.
33. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Detecting malicious domain names using deep learning approaches at scale. *Journal of Intelligent & Fuzzy Systems, 34*(3), 1355–1367.
34. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating deep learning approaches to characterize, signalize and classify malicious URLs. *Journal of Intelligent and Fuzzy Systems, 34*(3), 1333–1343.
35. Vinayakumar, R., Soman, K. P., Poornachandran, P., Alazab, M., & Jolfaei, A. (in press). Detecting domain generation algorithms using deep learning. In *Deep learning applications for cyber security*. Cham: Springer.
36. Vinayakumar, R., Soman, K. P., Poornachandran, P., Alazab, M., & Thampi, S. M. (in press). AmritaDGA: A comprehensive data set for domain generation algorithms (DGAs). In *Big Data Recommender Systems: Recent Trends and Advances, Institution of Engineering and Technology (IET)*.
37. Vinayakumar, R., Soman, K. P., Poornachandran, P., & Menon, P. (2019). A deep-dive on machine learning for cyber security use cases. In: *Machine Learning for computer and cyber security: Principle, algorithms, and practices*. Boca Raton, FL: CRC Press.
38. Vinayakumar, R., Soman, K. P., Poornachandran, P., Mohan, V. S., & Kumar, A. D. (2019). ScaleNet: Scalable and hybrid framework for cyber threat situational awareness based on DNS, URL, and email data analysis. *Journal of Cyber Security and Mobility, 8*(2), 189–240.
39. Vinayakumar, R., Soman, K. P., Poornachandran, P., & Sachin Kumar, S. (2018). Detecting Android malware using long short-term memory (LSTM). *Journal of Intelligent & Fuzzy Systems, 34*(3), 1277–1288.
40. Vinayakumar, R., Soman, K. P., Poornachandran, P., & Sachin Kumar, S. (2018). Evaluating deep learning approaches to characterize and classify the DGAs at scale. *Journal of Intelligent & Fuzzy Systems, 34*(3), 1265–1276.
41. Wang, S., & Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42*(4), 1119–1130.
42. Woodbridge, J., Anderson, H. S., Ahuja, A., & Grant, D. (2016). Predicting domain generation algorithms with long short-term memory networks. arXiv preprint arXiv:1611.00791.

43. Yadav, S., Reddy, A. K. K., Reddy, A. L., & Ranjan, S. (2010). Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (pp. 48–61). New York, NY: ACM.
44. Yadav, S., Reddy, A. K. K., Reddy, A. N., & Ranjan, S. (2012). Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *IEEE/ACM Transactions on Networking, 20*(5), 1663–1677.
45. Yu, B., Gray, D. L., Pan, J., De Cock, M., & Nascimento, A. C. (2017). Inline DGA detection with deep networks. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 683–692). Piscataway, NJ: IEEE.
46. Yu, B., Pan, J., Hu, J., Nascimento, A., & De Cock, M. (2018). Character level based detection of DGA domain names. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). Piscataway, NJ: IEEE.
47. Zhauniarovich, Y., Khalil, I., Yu, T., & Dacier, M. (2018). A Survey on malicious domains detection through DNS data analysis. *ACM Computing Surveys, 51*(4), 67
48. Zhou, Z. H., & Liu, X. Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering, 18*(1), 63–77.
49. Zhou, Z. H., & Liu, X. Y. (2010). On multi-class cost-sensitive learning. *Computational Intelligence, 26*(3), 232–257.