informatics

MDPI

Article

WordDGA: Hybrid Knowledge-Based Word-Level Domain Names Against DGA Classifiers and Adversarial DGAs

Sarojini Selvaraj and Rukmani Panjanathan *

Vellore Institute of Technology (VIT) Chennai Campus, Chennai 600127, Tamil Nadu, India; sarojini.s2015@vit.ac.in
* Correspondence: rukmani.p@vit.ac.in

Abstract: A Domain Generation Algorithm (DGA) employs botnets to generate domain names through a communication link between the C&C server and the bots. A DGA can generate pseudo-random AGDs (algorithmically generated domains) regularly, a handy method for detecting bots on the C&C server. Unlike current DGA detection methods, AGDs can be identified with lightweight, promising technology. DGAs can prolong the life of a viral operation, improving its profitability. Recent research on the sensitivity of deep learning to various adversarial DGAs has sought to enhance DGA detection techniques. They have character- and word-level classifiers; hybrid-level classifiers may detect and classify AGDs generated by DGAs, significantly diminishing the effectiveness of DGA classifiers. This work introduces WordDGA, a hybrid RCNN-BiLSTM-based adversarial DGA with strong anti-detection capabilities based on NLP and cWGAN, which offers word- and hybrid-level evasion techniques. It initially models the semantic relationships between benign and DGA domains by constructing a prediction model with a hybrid RCNN-BiLSTM network. To optimize the similarity between benign and DGA domain names, it modifies phrases from each input domain using the prediction model to detect DGA family categorizations. The experimental results reveal that dodging numerous wordlists and mixed-level DGA classifiers with training and testing sets improves word repetition rate, domain collision rate, attack success rate, and detection rate, indicating the usefulness of cWGAN-based oversampling in the face of adversarial DGAs.

Keywords: cybersecurity; Domain Generation Algorithms (DGAs); DNS; word-based DGA botnet; neural language models; RCNN-BiLSTM; conditional Wasserstein generative adversarial network (cWGAN); word embedding; natural language processing; command-and-control (C&C)

Citation: Selvaraj, S.; Panjanathan, R. WordDGA: Hybrid Knowledge-Based Word-Level Domain Names Against DGA Classifiers and Adversarial DGAs. *Informatics* **2024**, *11*, 92. https://doi.org/10.3390/informatics11040092

Academic Editor: Olga Kurasova

Received: 21 June 2024
Revised: 30 September 2024
Accepted: 14 October 2024
Published: 26 November 2024

1. Introduction

Cybercriminals commonly use malicious botnets to monitor network traffic, send malicious emails, construct Domain Generation Algorithms (DGAs), steal data, and carry out Distributed Denial-of-Service (DDoS) attacks [1]. Attacks like this aim to give the botmaster a stable connection to control compromised bots by sending them to a Command-and-Control (C&C) server. The infected bots communicate with the C&C server using domain names. Botnets are blocked by adding their domain names to a blocklist [2]. Fraudsters use DGA technologies to amplify the obfuscation of the C&C server. In other words, a C&C server's IP address registers a single AGD in the DNS. Although trojan assaults have been extensively explored in various applications, including natural language processing (NLP) [3], few studies have been conducted on DGA detection. Consequently, as a research problem, cybercriminals are now thinking about how to execute a trojan attack on a deep learning-based DGA detection system.

Meris [3], one of history's most potent botnets, carried out the most significant HTTP attack and rallied to launch more critically destructive DDoS attacks in 2020 [4]. This was led by a botnet coordinating 402,000 diverse IP addresses, illustrating the intensity of botnet-enabled attacks. A C&C communication channel is typically used by remote attackers,

also known as botmasters, to command many malware-infected devices or "bots", which make up a botnet [3]. Individual bots typically use a DNS to locate C&C servers by translating their domain names into IP addresses. More recently, botnets have improved their stealth capabilities by using DNS domain fluxing to avoid discovery. Domain Generation Algorithms (DGAs) are used in DNS domain fluxing to prevent block listing.

Defense efforts become much more challenging if the attacker raises the rate at which domain names are generated and modifies the domain name for the C&C server. Domain fluxing is the term used to describe this type of evasion of DGA implementation [3]. AGDs are typically short-lived owing to domain fluxing, and they connect several domain names registered by DGAs to the IP addresses of the C&C servers. However, blocking or removing dangerous domain names typically takes time due to the coordination required between DNS registrars and ISPs.

DGAs [3] generate a set of pseudo-random domain names, allowing the C&C server's domain name to vary regularly. After using the seed to create a list of domain names, the botmaster chooses a portion of unregistered domain names to register. Bots can generate a collection of domain names using an identical DGA and seed, then query each individually until they can connect through the C&C server. DGAs can produce several hundred domain names daily; they must locate these names and investigate to destroy the botnet. The first objective highlighted in recognizing the algorithmically generated domain (AGD) was to gather binary samples of bots, retrieving their methodologies and seeds, and constructing domain names before mitigation [5].

An attacker can generate many pseudo-random or algorithmically produced domains by incorporating a DGA into the bot program. With a vast AGD collection, the attacker's C&C communication is limited to a single domain name. We investigated novel DGAs that maintain their anti-detection abilities while reducing the acute reliance on detector information to achieve significant anti-detection ability and practicality. Much research has recently been conducted on the automatic domain name classification approaches produced by DGAs, most of which have centered on the character- and word-level detection of potentially harmful domain names. For instance, it is more difficult to identify malicious domain names produced by word-based DGAs [5]. The use of adversarial DGAs to generate domain names is becoming more widespread, and these names highlight some of the shortcomings in the DGA classifiers that are now extensively employed.

Accordingly, these adversarial DGAs cannot highlight the fundamental shortcomings of current character-level-based DGA classifiers or indicate critical directions for their improvement. As computer hardware has advanced, deep learning (DL) has gained popularity among machine learning (ML) approaches. Using deep neural networks to solve high-accuracy classification problems has proven successful. To that aim, we describe a procedure to generate realistic botnet data by employing oversampling techniques and generative adversarial networks (GANs) to enhance the ability of classifiers to identify possible evasion scenarios. Recent research [6–10] has demonstrated the remarkable effectiveness of GANs [10]. To learn their evasions, we propose utilizing 'precision' as the criterion for measuring generator performance in an AUC-Pre, rather than 'accuracy'.

Due to their ability to replicate intricate probability distributions, GANs are used in data oversampling [4]. Although artificial oversampling techniques like SMOTE and ADASYN [11] address the low-data scheme, they are inadequate when used with data with complicated and high-dimensional probability distributions. The abovementioned datasets are split into ten folds to evaluate post-augmentation generator performance, with a 70–30 train–test split. All six classifiers' F1 scores, accuracy, recall, and precision were used to assess the botnet detectors' performance following data augmentation.

In these studies, datasets such as OSINT, DGArchive, Bader's, 360NetLab, UMUDGA, UTL_DGA22 dataset, or Tranco domain are utilized to assess the efficacy of their solutions regarding DGA botnets. New families and variants of DGA botnets emerge over time. This implies that the DGA botnet datasets must also be updated regularly for new botnet families. We should simultaneously increase the number of botnet family samples that

are now available. As a result, the datasets become more realistic, enhancing the ability to evaluate recently suggested techniques.

To attain the maximum accuracy value, feature selection relies on innovative techniques. To achieve the highest level of accuracy in both classification tasks, word embedding, TF-IDF, FastText, and base features should be combined.

Wordlist-based combinations, such as domain name readability, have replaced random alphanumeric letters in DGA. Researchers suggested using GAN-based DGAs, including DeepDGA [6] and Khaos [7], to provide better domain names. DeepDGA is an initial effort to create domain names using standard character-based GANs. By merging a robust GAN variation called Wasserstein GAN (WGAN) with an n-gram vocabulary, Khaos enhances DeepDGA. AGDs with strong anti-detection capabilities are generated by DGAs utilizing a GAN-based model. Because the GAN-generating process is unregulated, GAN-based DGA produces many repetitive and conflicting domain names that violate domain name constraints, limiting GAN's practical applicability. As a result, research into creating adversarial samples to test deep learning models for DGA detection is significant. Simultaneously, to effectively counteract adversarial sample attacks, we use the conditional adversarial generation network to generate identical data with varied noise and labels, which may then be used to replicate traffic patterns caused by malware variants.

This research proposes a domain generation method known as WordDGA. This wordlist-based evasion technique first trains a backdoor adversarial assault on a hybrid DNN model-based adversarial DGA detector with high anti-detection capabilities. It trains a conditional Wasserstein generative adversarial network (cWGAN) using a Gumbel softmax distribution with a generator and discriminator. To that end, this study chose to use a cWGAN for WordDGA.

The proposed WordDGA uses NLP techniques like n-grams ($n = 3, \ldots, 7$) and word2vec to train an autoencoder. This ensures that AGDs follow vocabulary language norms and have strong anti-detection capabilities on word- and hybrid-level DGA classifiers. The actual samples are encoded using a pre-trained hybrid DNN model to generate a training dataset for the GAN generator, and the GAN's parameters are then updated by oversampling while training the generator and discriminator. The experimental findings highlight the need for robust DGA classifiers, since they demonstrate that the AGDs produced by WordDGA defy the ideal backdoor attack efficacy model and result in domain names that are challenging for people to understand. The suggested approach may be effectively applied in real-world industrial contexts, as demonstrated by the investigations into the capabilities of anti-detection, word repetition rate, domain collision rate, and assault success rate, as well as comparisons with other research DGAs.

The remaining contributions are summarized below:

- To develop a text-based technique for generating domain names for NLP tools, including word distribution, tokenization, and integration with the DGA botnet. The goal is to improve wordlist-based DGA identification by establishing a domain with relevant English words and providing statistical and sequence-dimensional features.
- The initial WordDGA model includes adversarial DGA semantic subsequences between words based on benign and DGA domain names. Domain names alter legal DGA production to keep DGA enemies from being detected.
- To develop adversarial and oversampling approaches for generating genuine domain names using cWGAN. They developed DGA domains with anti-detection features. Learn how detectable and adversarial DGAs can increase WordDGA's repetition and collision rates.
- The research for WordDGA's effectiveness of wordlist-based DGA botnets constructed using actual domain retraining should be carried out and approved. Adversarial DGA has an accuracy range of 98.72–99.01%. The model achieves a 98.63% accuracy rate in word DGAs and mixed DGA dataset detection, surpassing earlier techniques.

The remainder of the tasks are organized as follows: Section 2 discusses the literature survey; Section 3 explains the methodology for the proposed detection-based wordlist

DGA botnet mechanism, which is based on deep learning and oversampling techniques; Sections 4 and 5 discuss the experimental design and results; and Section 6 concludes with future research.

2. Literature Review

In recent years, researchers have worked on the DGA domain to develop and explore the different methods that have been suggested for these adversarial networks and DGA identifiers with encouraging outcomes. These methods may use identifiers beforehand to evaluate the robust anti-detection capabilities of DNN with GAN oversampling techniques in our experimental investigations.

Research was undertaken on the succeeding generation of DGAs, which were dissimilar before being classified. The corresponding development of common DGAs may be separated into the hash, arithmetic, wordlist, and mixed-predicate DGA methods [12]. In a wordlist comparable to Matsnu, Gozi, Pizd, and Suppobox have extremely rigorous procedures before ascertaining and producing the domains through reconnecting more phrases in the word DGA. Even though the mixed-predicate approach is comparable to VolatileCedar, it has remarkable DGAs and generates domains through a comparable domain [3]. This section categorizes some recent prior work that may be relevant to classical-based ML and DL classification based on ML-based techniques, such as [13–15]. Most of these wordlist-based identification procedures are listed in this section, and the methods have domain-based detection. Table 1 compares DNN-based methods to recognize security attacks in the DGA botnet recognition mechanism.

Table 1. DGA-based botnet families and illustrations of DGA domains.

Botnet Families	DGA Scheme	Domain Name Samples
Bigviktor	Word-based	knowredpermit.art winstilllandscape.club helppurpledistance.fan
Matsnu	Word-based	row-closed-bid.com brushpot-guide.com chickenpriceresearch.com
Suppobox	Word-based	necessarypower.net pleasantcountry.net groupfirst.net journeythird.net
Banjori	Mixed	ztgxrasildeafeninguvuc.com igxbtallulahavaw.com umdhrasildeafeninguvuc.com
Nymaim2	Word-based	Smilefavorite.uz Soft-professor.com Sculpturenegative.net
Pizd	Word-based	advanceprepare.net distantmanner.net desiregentleman.net animalwater.net
Gozi	Word-based	andestablishment.ru formsworkfreeall.com
Kfos	Word-based	Help-google.tw
Ngioweb	Word-based	Subozirion-multirusenelike.com
VolatileCedar	Mixed	Shplayergetadobaefl

The work uses hybrid deep learning techniques, such as RNN, CNN, and BiLSTM, with oversampling methods to build identification models for classifying benign domain

names. The domain is produced and applied through the wordlist-based DGAs, and the model of the theoretical extracts emphasizes several domains that contain the features of statistical *n*-grams and domain word distribution features. The experimental results show that most hybrid deep learning-based models with oversampling methods produce good results. The previous model could discover many categories of DGAs, but these conditions still have significant weaknesses. It may not have found the word or mixed-predicate DGAs. DL-based DGA classifiers were contrasted with the manual extraction of highlights in ML. Deep learning networks can perform better than DGA detectors, such as CNN and LSTM, which extract features from data to achieve automatic feature extraction. Deep learning-predicated DGA detectors perform better than machine-learning-based DGA detectors. The consequence of these intrinsic illustrations of DGA is the detection of wordlist-based tasks that present demanding tasks for classifiers. We summarize some techniques from recent studies, datasets, taxonomy, and classification results by considering the methods, approaches, evaluation setup, and classification results shown in Table 2.

Table 2. A summary comparison of featureless DL methods by these severely proper modern techniques of DGA predicated botnet.

Reference	Year	Dataset (Benign/Botnet)	Model
Woodbridge [16]	2016	Alexa, Bambenek	LSTM
Lison and Mavroeidis [17]	2017	Alexa, DGArchive, Bambenek	RNN
Koh and Rhodes [18]	2018	Bader, Abakumov	LSTM
Tran [19]	2018	Alexa, Bambenek (37)	LSTM.MI
Vinayakumar [20]	2018	Alexa, OpenDNS/Bambenek, Bader	LSTM, GRU, IRNN, CNN, CNN-LSTM
Xu [21]	2018	Alexa, DGArchive	CNN-based
Yu [22]	2018	Alexa, Bambenek	LSTM, BiLSTM, Stacked CNN, Parallel CNN, CNN-LSTM
Akarsh [23] and Qiao [24]	2019	Alexa, Bambenek	LSTM
Liu [25]	2020	Alexa, Netlab, Bambenek	Hybrid (CNN-BiLSTM)
Ren [26]	2020	Alexa, Bambenek, Netlab	CNN, LSTM, CNN-BiLSTM, ATT-CNN-BiLSTM, SVM
Cucchiarelli [27]	2021	Alexa, Netlab (25)	BiLSTM, LSTM.MI, CNN-BiLSTM
Highnam [28]	2021	Alexa, DGArchive (3)	CNN-LSTM-ANN
Namgung [29]	2021	Alexa, Bambenek	CNN, BiLSTM, CNN-BiLSTM
Yilmaz [30]	2021	Majestic, DGArchive	LSTM
Yang [31]	2022	Alexa, Netlab360 (9)	Sub word tokenization and transformer
Vranken and Alizadeh [32]	2022	Alexa, DGArchive	SVM, MLP, RF, DT, KNN
Ran and Fong [33]	2023	Tranco, UMUDGA	SubWord-CNN and SubWord-LSTM

In recent adversarial DGAs [6–11], a standard attack on evasion DGA classification has achieved the desired result. These existing works shown in the current machine and deep learning-based DGA classifications have achieved better execution in numerous aspects and the face of adversarial samples. The optimization of adversarial DGAs has received a lot of attention in recent adversarial learning research. DeepDGA [6] first combined GAN with botnet approaches to prevent DGA identifiers. Khaos [7] is an adversarial DGA with strong anti-detection abilities based on DNN approaches, and the WGAN synthesizes DGA to produce invisible domains that detect lower detection rates through traditional DGA approaches using the distribution of *n*-grams and domain pronounceable rates to generate DGA through the WGAN. DeceptionDGA [8] directly uses statistical features to detect legitimate domain names and effectively avoids FANCI detection to reduce the rate in DL, such as the LSTM.MI classification. By changing the values of the linguistic highlights produced by ML techniques, DeceptionDGA [8] aims to avoid that. The adversarial DGA, which was discovered by the Deception and Deception_2 DGA-based botnet using a white-box attack based on machine learning, avoids detection;

however, this deteriorates with random forest classification and produces worse results in DL classifiers that need feature knowledge that is manually created using the FANCI [5]. MaskDGA [9] presents adversarial learning approaches to avoidance technology against text characterization models, identifies the half-character of domain names, and uses known vulnerability classification for inline detection. It also uses a modular extension of DGA to make it practical for adversaries. It computes the Jacobian saliency matrix, or JSM, based on a particular DGA classifier starting at 97%. MaskDGA resulted in adversarial samples outperforming the DeepDGA and achieved an adversarial effect against four DL-based DGA models.

CLETer [10] influences the current DGA domain of each character of the classification results and creates the character to modify adversarial examples. The DGA categorization has improved anti-detection abilities and effective malware awareness. Consequently, CLETer proposes that the DGA performs significantly better in detection than the natural character-based DGA disturbance alone. This may be used to all DGA categories, including the setup of tools for thorough detection.

3. Background

3.1. Word Extracting in Domain Name

To prompt the domain name based on characters and words, they are converted to the fixed fields to simplify the GAN with oversampling techniques, and the approaches of specialists are detailed as follows:

A sample of the primary section of the domain names, such as the feasible top-level, secondary, and territory domain name sections, are typically irregularly or consistently from a particular registry to the extraction process. This sample must match the host and server label in DNS. It is acceptable to omit the section that displays the server's name for these domain names. The subsequent section is utilized for integral conversion when the word is supplied; the domain name's words are then changed to compare the word embedding; and an equally amazing transformation into the digital vector occurs as a result of the analysis of the word embedding's distinct position. Lastly, the mapping domain's location validates that code handling may be carried out by reversing the decoding. Text or strings cannot be used as data in deep learning models [34]. Thus, the vectorization of the incoming data is required. Word embedding can vectorize the extraction results, the domain's *n*-gram sequences, into word vectors [35]. With the use of libraries like Word2Vec [15], FastText [16], or GloVe [17], word embedding techniques have been developed to help with word-embedding tasks. GAN is utilized in the current field of text handling, similar to the categories of text and images, the awareness of word features [19], and dealing with usually uninterrupted information. DNN handles discrete word-based DGA families, which typically involve investigating theoretical results.

The primary structure of the system suggested in this study is shown in Figure 1, and it is based on a hybrid approach of RCNN and BiLSTM and the cWGAN technology. Data preparation is a crucial aspect of data analysis that directly impacts prediction accuracy. The data preparation module performs actions on the original data to make it more suited for the model's prediction, such as one-hot encoding and data normalization. We presented cW-GAN [36], a novel approach that integrates over-sampling to produce a perfectly balanced dataset and a detection model based on *n*-gram sequences. A global-average-pooling layer reduces the number of parameters, a dropout layer deactivates the network connections, and the detection results are obtained using Gumbel softmax [37] after activation with the Relu layer. It delves thoroughly into the location and text characteristics of malicious domain names.

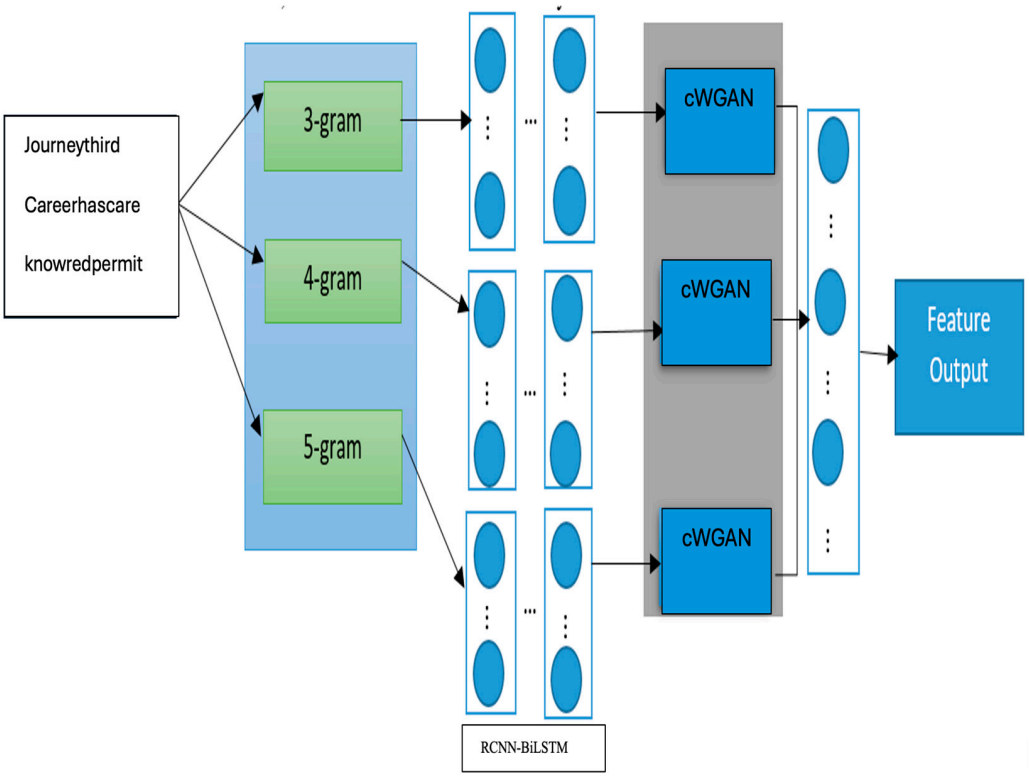


Figure 1. Model structure.

3.1.1. Data Preprocessing and Normalization

The top-level domain (TLD) is removed from the domain name during data preprocessing, and the remaining parts are mainly used as the category's fundamental unit. Meaningful word segmentation [38] is accomplished with Wordninja, and *n*-gram sequences for domain names without TLDs are produced using *n*-gram algorithms. We create 3–7 g sequences.

The extraction of features is the next step. To derive significant word segmentation attributes, we initially calculate context-sensitive word-embedding features using ELMo [39] and aggregate the word features to generate 128-dimensional word distribution features. Then, we extract 11-dimensional statistical features from the domain name and segmented words and translate them into 8-dimensional statistical features. The BiLSTM model is used to construct the 128-dimensional 3–7 g sequence features for *n*-gram feature extraction, depending on the outcomes of 3–7 g sequences. First, we use Wordninja [40] to extract the word sequence of the domain name. Next, we use the ELMo approach to generate a 1024-dimensional context-sensitive word-embedding vector of features for each word. Words in legitimate domain names typically have some semantic correlation. However, the dictionary-based AGDs [41] of randomly concatenated words have comparatively little semantic correlation. As a result, we detect dictionary-based AGDs by analyzing word correlation [42].

Given that the RFC standard of the DNS states that a domain name's maximum label length is 63 and that the unidentified string of an AGD is frequently seen at the initial label of a website's URL, we fixed the sequence length to 64. Finally, we extract sequence features based on the sequences produced by 3–7 g [43], using the BiLSTM model to output 128-dimensional features independently.

We used well-known oversampling techniques to balance the dataset as a preprocessing step in our methodology. The following stages of data preprocessing are normalization and dataset splitting. These are elaborated in the following sections.

The network traffic characteristics and 11-class labels in the training, validation, and test sets were converted into several formats to facilitate computation and improve con-

vergence throughout the DL model construction process. Features can be divided, labeled, reshaped, and normalized in the data preprocessing stage [28]. Using the min–max normalization technique provided by Equation (1), each value of the network traffic feature was first scaled to a range between 0 and 1:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

where x is a feature vector representing network traffic, and x_{min} and x_{max} are x's minimum and maximum values, respectively. To facilitate the creation of the DNN model, an additional dimension (t = 1) was added to the feature set to represent a single time step. The dimension of the feature set is modified from $X \in \mathbb{R}^{p \times q}$ to $X \in \mathbb{R}^{p \times 1 \times q}$, where p is the total number of samples and q is the total number of features. In the label vector, integers 0–10 represented the numerical values of the 11 classes.

To train and assess the resilience of deep learning models against underfitting and overfitting, the entire collection of network traffic data is divided into training sets (70%) and test sets (30%). This work addressed overfitting in the neural networks using the hold-out validation technique. The model training process utilized the testing set samples to adjust the hyperparameters. Furthermore, the test sets' previously unobserved network traffic samples were used to assess the DL models' generalization capacity.

We first handle the dataset with the Python version 3.5 tldextract module, preserving the top-level domain and subdomain for the second-level domain of the website's names. Next, the word-hashing algorithm is used to add an analogous flag bit (such as "#") to the start and finish of each string. Although adding the sign bit results in more word-hashing vectors after n-gram [43] processing, this process enhances the dimensionality of the retrieved features, because the extra word-hashing vectors reveal the beginning and ending levels of the character instances in the domain names.

3.1.2. Domain Name Encoding

A domain name's character sequence can be encoded using the domain name encoding module. Initially, we create a character dictionary by using the recurring frequency of every word in the domain name dataset. We then assign a unique number to each character depending on its frequency of occurrence. Next, a domain name is represented as a character and word sequence, with its initial vector representation being $v' \in \mathbb{R}^l$, where l is the domain name's upper bounded length, the ith element of v' is the actual number of the unique character, and the word dictionary assigns it to the ith character in the character sequence. The last step is to transfer each element of v' to the final fixed-length vector representation $v \in \mathbb{R}^{l \times d}$ of a domain name.

3.1.3. Split Dataset

We have trained and assessed our models for 100 runs due to the limited number of samples in the datasets, making the classification result exceedingly unstable. We randomly shuffle and divide the data into training and testing sets for each run. We then train the sets using the model for 2000 epochs and assess the results.

3.1.4. The Theory of GAN

A higher classification of likelihood could provide a bad classification result. The adversarial DGA example is a fine-tuned cross-section created by enlarging the real invisible example. So, the DNN is exposed to GAN models. GAN requires a vector of two DNN mechanisms; RCNN is fully connected by affiliated neurons of the same double layer, BiLSTM, and the final activation layer uses Gumbel softmax distribution. Additionally, pre-trained decipherers compute the findings prior to the farthest level, and domain generates the outcomes of that deciphering. The activation layer's position is determined by Gumbel softmax, which has a falsified loss function. In these connected domains, we generate the

domains designing by the invariant sequence to the sequence of the inferior extent of the integral training gap. The encrypt p before creating suspicious fields is the discriminator.

3.2. Models

This section presents the two Artificial Neural Network (ANN)-based models that we have proposed: The Recurrent Convolutional Neural Network (RCNN) and the Bidirectional Long Short-Term Memory Network (BiLSTM).

3.2.1. RCNN

To obtain the forward and backward context for a domain name represented as $v \in \mathbb{R}^{l \times d}$, RCNN uses a bi-directional recurrent structure. The forward and backward context of each character is defined as follows:

$$C_l(v_i) = f_1(W_l C_l(v_{i-1}) + W_{sl} e(v_{i-1})) \tag{2}$$

$$C_r(v_i) = f_1(W_r C_r(v_{i+1}) + W_{sr} e(v_{i+1})) \tag{3}$$

where f_1 is a nonlinear activation function, W_l, W_r, W_{sl}, and W_{sr} are weight matrices, and $C_l(v_i)$ and $C_r(v_i)$ are the left and right contexts of the character v_i, respectively. The character embeddings of the former character v_{i-1} and the latter character v_{i+1} are represented by the letters $e(v_{i+1})$ and $e(v_{i-1})$, respectively. The left context of v_i is obtained from the left context v_{i-1} and the character embedding, whereas the right context v_i is derived recursively from the right context of v_{i+1} and the character embedding. Once the characters and the right contexts have been obtained, the left, right, and initial character embeddings of v_i are cascaded together to create the output feature vector of x_i.

$$x_i = [C_l(v_i); e(v_i); C_r(v_i)] \tag{4}$$

A feature representation can express domain names as $x_d = [x_1, x_2, \ldots, x_l]$ once the latent feature vectors for each character have been obtained in the name.

The convolution procedure is then carried out on x_d using a convolutional filter of size $h \times d$ after x_d has been fed into the convolutional layer, as indicated below:

$$o_i = F(w_1 \cdot x_d[i : i + h - 1]) \tag{5}$$

$$c_i = f_2(o_i + b_1) i = 1, 2, \ldots, s - h + 1 \tag{6}$$

$$c = [c_1, c_2, \ldots, c_{s-h+1}] \tag{7}$$

where o_i is the output of the convolution process, b_1 is the bias term, w_1 is the weight matrix of the convolutional kernel, f_2 is the ReLU activation function, and F is the convolutional size $h \times d$ filter. Together, these elements comprise the feature map c of an input domain name. Additionally, $c_i|_{i=1}^{s-h+1}$ is extracted with the local feature of the convolutional kernel.

3.2.2. BiLSTM

To obtain the contextual characteristic of every character over a large distance, we employ BiLSTM. It can learn and capture intricate connections between many characters inside sequential data, regardless of their distance. Two layers of hidden nodes from two distinct LSTMs make up the bidirectional architecture. The two LSTMs capture dependencies in different directions.

While the activation in the second hidden layer passes backwards in the sequences, the first hidden layer has recurrent connections from the final syllables. We may, therefore, obtain the forward hidden state from the forward LSTM network and the backward hidden state from the backward LSTM network at the LSTM layer. The compositional semantic information is captured by the two states in both directions of the character sequences. There is a forward and a backward LSTM in the BiLSTM. The CNN layer's output vector is $c = \{c_1, c_2, \ldots, c_n\}$. The forward LSTM reads the input vector from $\{c_1\}$ to $\{c_n\}$, and the

input vector is read by the backward LSTM from $\{c_n\}$ to $\{c_1\}$; that is, in reverse order. In the meantime, two hidden states are generated: $(\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_z})$ and $(\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_z})$. By combining the two hidden states, as shown in Equation (4), we may obtain the BiLSTM layer's output.

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \tag{8}$$

The feature-based paradigm has several drawbacks, including the need for additional natural language processing tools, a high impact on performance from accumulated errors, and insufficient feature extraction.

3.3. Statistical Feature Extraction

The word features' mean and standard deviation are considered the final word distribution features during extraction. Additionally, we suspect the statistical values of segmented words and the distribution of parts of speech are also significant. Consequently, we also extract statistical features to complement the deep learning features mentioned above, enriching the feature types and enhancing the detection result after meaningful word segmentation. From three viewpoints, namely domain length features, character ratio features, and word statistical features, we extract 11-dimensional statistical features.

As the four dimensions of the statistical features—the number of words, the average word length, the length of the domain name without TLD, and the length of the original domain name—have integer values, we normalize them by dividing them by the highest value found in the dataset. Additional normalization of the remaining seven-dimensional characteristics is unnecessary because their values already fall within the interval [0,1]. The 11-dimensional statistical characteristics are then converted into 8-dimensional features using a fully connected layer. Table 3 displays the 11-dimensional statistical features in detail.

Table 3. Selected word distribution features on DGA domain name classification.

Feature Name(s)	Description
DomainLength	Domain token count
NoOfDomainWords	Number of words in domain
NoOfHypen	Number of hyphens in domain
RareRatio	The ratio of rare words in domain
CommonRatio	The ratio of common words in domain
RatioMisspelled	The rate of misspelled words in the domain name
NoOfRandom	Number of random domain words
Vowel_freq	Number of vowels within the domain
Reputation	Number of *n*-grams in allowlist
Words_I	The real integer of words behind the substantial word distribution
JJ_Ratio	The rate of adjectives in a word behind the substantial word distribution

3.4. DGA DNN Detection Algorithms

(1) RNN: The structures of RNN capture information about sequences and use different types of structures. This structure learns word sequences created through domain names, and the network has a more significant time step that can increase the possibility of RNN due to issues concerning the error gradient and vanishing with bidirectional architectures.

(2) CNN: CNNs were initially designed for tasks regarding text and image generation with well-recognized success [19,20]. The extracted features by the convolution layer and pooling are typically fed to a fully connected layer that performs prediction. When CNN is used to create the input map of features that display the combination data features performs better than character-based Keras [34] and word-based word2vec [18] embeddings. CNN

is used in both character- and word-level classification. Lastly, the final pooling layer is replaced by a fully connected layer for classification in various DNNs passed through the word-based domain name sequence.

(3) Bidirectional LSTM (Bi-LSTM): Bi-LSTM is an extension of LSTM. LSTM only processes the sequence in one direction, whereas Bi-LSTM incorporates the information in both directions. To achieve this, Bi-LSTM adds one more LSTM layer to the standard LSTM, which reverses the information flow direction of the other LSTM layer. The bidirectional information flow can be combined to produce a more meaningful output. So, Bi-LSTM has advantages over LSTM in many real-world problems. The Bi-LSTM model was proposed in [20] to address the AGD detection problem.

(4) RCNN-BiLSTM (hybrid-based): RCNN-BiLSTM [25] is intended to combine the benefits of RCNN and BiLSTM. It applies an RCNN layer to the input as a feature extractor. The output feature map, which is also sequential data, is then processed by a BiLSTM layer. The design is based on the comparative capabilities and limitations of RCNN and BiLSTM. RCNN is more suitable for capturing local information because it lacks the mechanism of recognizing global information over long distances. By contrast, BiLSTM does better at extracting global information thanks to its memory mechanism.

4. Proposed Model

However, it has been noticed that the model of the proposed detection of DGA connects two model features, namely the character-based [18] and wordlist-based [35] DGA models. The figure illustrates the two stages of training and testing. This ensemble begins by adjusting the edges towards both DNN DGA based on GAN with oversampling techniques, enabling an effective identification of both character- and wordlist-based DGAs. It is displayed in Figure 2, and all model features are informed independently of the training phase. The pre-processing, data extraction, highlight, and training stages are directed toward the method of word-based DGA, and the word-based model is trained as a hybrid DNN network with oversampling strategies. These distinguish between the hybrid DNN for solving problems, so that it is effective in classification [35].

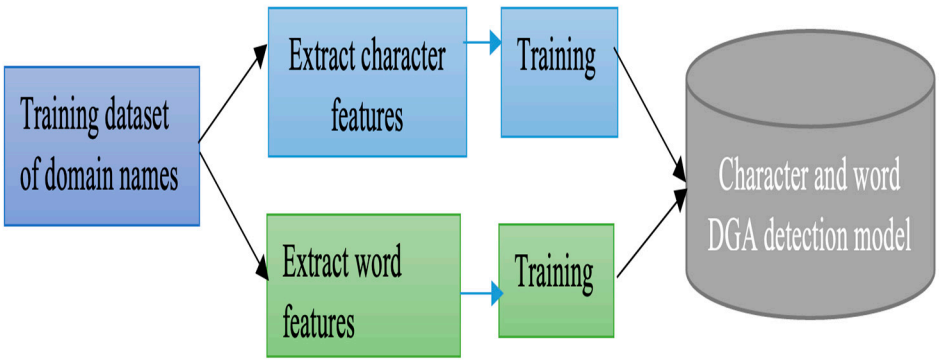


Figure 2. The training process of the proposed DGA detection botnet model.

The phase of testing for the model features of character-based and word-based DGA detection is used to handle each monitored DNS separately, as shown in Figure 3. For the proposed word-based DGA detection, the data pre-processing step and the procedures of each category for each DNS observed have been concluded. The model utilizes this information to specify connections between the most distinct examples of awareness with a wide range of these maximum hybrid approaches, determining the average of fully accurate predictions for the distinctive characteristics of both models.

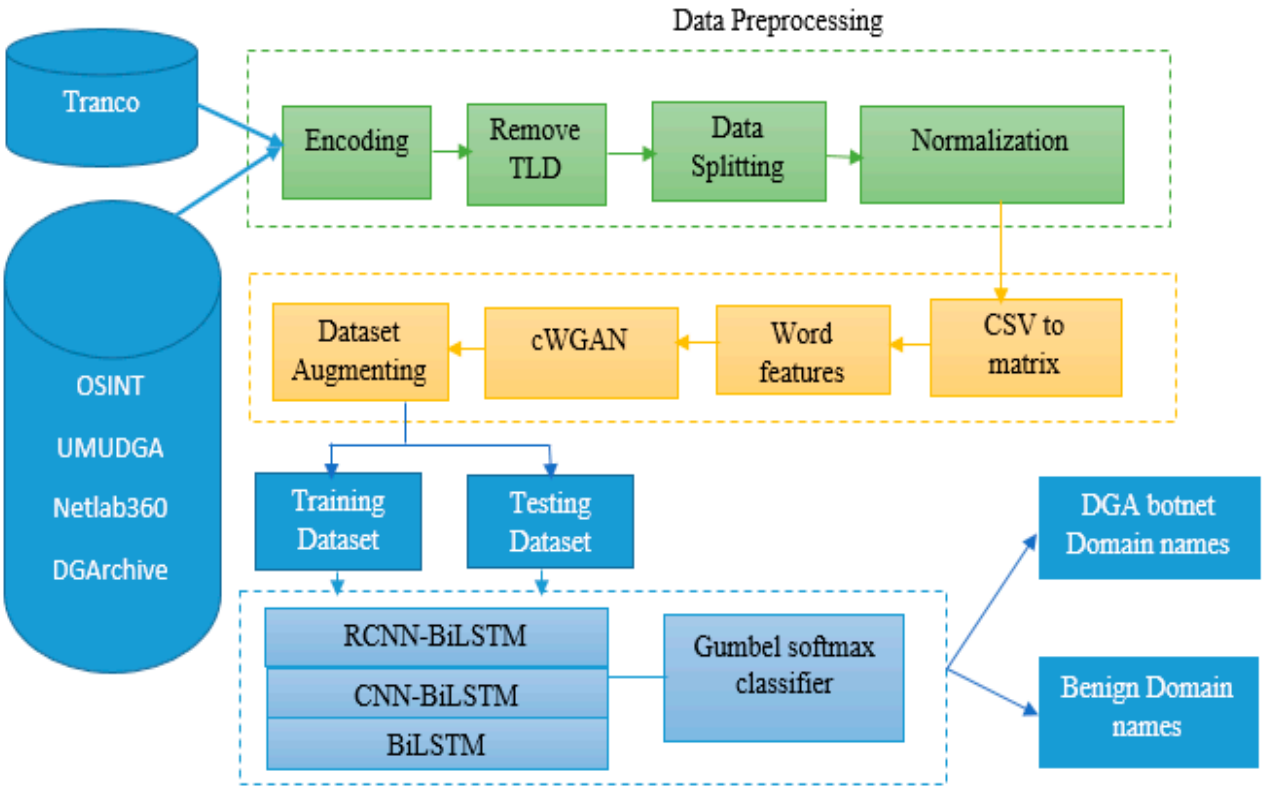


Figure 3. The structures of the proposed model.

Figure 3 displays an overview of the system architecture suggested in this study to identify malicious domain names based on DGAs employing cWGAN-based RCNN-BiLSTM and word embedding.

This paper's model architecture comprises four parts: the output layer, the RCNN-BiLSTM layer, the cWGAN, and the word embedding. To avoid overfitting, which is learned before it reaches the output layer, the domain-name sequence uses dropout [25].

The performance of the discriminator networks in DGA domain names and the categorization of the adversarial DGA strings synthesized from the generator networks are the two expected outcomes of GAN research that are set up for evaluation in order to assist generation. The basic GAN model and three GAN variants of WGAN-GP [36], cGAN [4], and cWGAN-based oversampling [5] are exploited for each investigation.

They use DGA, valid domain names from datasets, and pre-trained *n*-gram embedding. Figure 4 illustrates the pre-trained *n*-gram embedding model creation procedure to provide clarity.

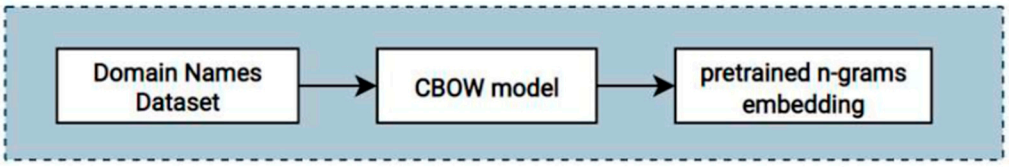


Figure 4. Pretrained *n*-gram embedding.

According to this study, domain names are equivalent to word sentences. In this context, "words" refer to streams of *n*-grams at the character level. The word embedding technique FastText Continuous-Bag-of-Words (CBOW) was used, since it performs well in text categorization [26,27]. This study found the optimal number of n for DGA domain detection using 3 g, 4 g, 5 g, 6 g, and 7-g samples. The input sequence's domain name length is set to 70, and the embedding size is set to 100. The RCNN-BiLSTM layer input will be the result of *n*-gram embedding.

4.1. GAN

4.1.1. The Conditional GANs

By using class labels to condition the model learning process and enable the multi-modal production of associated attributes of $P_{data}(x)$, this GAN [38] extends the Vanilla architecture inputs to include auxiliary/extra information (y). The two-player minimax objective function is a modified version of the normal GAN loss function that incorporates conditional probability for both G and D.

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[\log D(x|y)] + E_{E_{x\sim P_z(z)}}[\log(1 - D(G(z|y)))] \tag{9}$$

The design was nevertheless plagued with mode collapse and vanishing gradients, although this improvement enhanced generation and training stability.

4.1.2. The Wasserstein GAN (WGAN)

The Wasserstein architecture calculates the Earth Mover (EM) distance by looking at the difference between the quantity moved and the probability distributions of $P_{data}(x)$ and $G(z)$. The term Wasserstein loss (W-Loss) is used to describe this loss. The architecture swaps out D for a Critic (C) that generates a score instead of the probability of the data distribution by using a linear output activation function (instead of the sigmoid function). The objective function was created by [43] by adding a penalty term, as follows:

$$E_{x\sim P_{data(x)}}[C(x)] - E_{\hat{x}\sim P_g}[C(\widetilde{x})] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[(||\nabla_{\hat{x}} C(\hat{x})||_2 - 1)^2\right] \tag{10}$$

where $P\hat{x}$ is the sample of the uniform data point ($\hat{x}$) between P_{data} (x), and x are the data from P_g, the generated distribution. In addition to employing W-Loss to handle the mode collapse and vanishing gradient issues, the coefficient parameter guarantees that it prevents convergence failure.

4.1.3. cWGAN-Based Oversampling

The initial sequence of the GAN and the final sequence that emerges from this data oversampling will be assessed in advance of the dissimilarity dataset being oversampled. First, the GAN will generate more samples for the minority label, which will subsequently be integrated into the cGAN framework. This will make it possible to evaluate conditional pA | b sequences linked to the particular minority category sample layers. This will guarantee that the model produces additional training layers for the minority label a_n = G(c, b = b^ minority).

The WGAN employs a casualty posture that combines the experimental and conceptual advantages of the previous one in order to attract the interest of a distinct category label. In order to materialize the alternatively utilized class with straightforward, realistic objectives used in the label, the conditioning procedure augments the generator's lack of conditional incentive in producing insufficient samples, which promotes the extension of the absent position. This allows us to use the conditional loss while also conditioning the discriminator, which is impossible with the original cGAN position, with these parameters among those sharing D and G.

Thus, we have the following:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \underbrace{E_{a\sim p_{data}}[\log D(a)] - \mathbb{E}_{z\sim p_z}[D(G(z))]}_{wasserstein\ loss} + \underbrace{c\mathbb{E}_{z\sim p_z}\left[CCE\left(-\frac{1}{2m}\sum_{i=1}^{m}\log D(a_i,b_i) + \sum_{i=1}^{m}\log \frac{(1 - D(G(z_i,b_i),b_i))}{-\frac{1}{m}\sum_{i=1}^{m}\log D(G(z_i,b_i),b_i)}\right)\right]}_{Conditional\ loss} \tag{11}$$

where c presents the conditional GAN classification, and the *CCE* presents the category of cross-entropy between the label of the actual layer $b \in \{0,1\}$ and the likelihood of predicated class $b_{pred} \in (0,1)$, defined as the $CCD(b,b_{pred}) = -(b \log(b_{pred}) + (1 - b) \log(1 - b_{pred}))$, which presents the complete integer of samples, with the unique series preferred by the conditioning of the model; a presents the actual data and b presents the additional labels for the conditions with GAN, known as conditional GAN. The batch size has similar actual and noise data increments with G and D independently.

In terms of significance for the WGAN [36], by conditioning fluctuating balance, the loss of the conditional through the balance of components, which has been frequently altered by a certain percentage to the unconditional rate of $D(G(z))$ through the collection of standard $\lambda_{AC} = 0{:}1 \mid D(G(z)) \mid$, ensures that the Wasserstein loss lowers the objective of the main generator. The collection of standard backpropagation serves as the constant, and that denotes that, when the generator prompts examples of penalized have certainly detectable as effects of the layer of the conditional in the GAN. In the final layer of the generator, the distribution of Gumbel softmax is used to replace the block of residual layer. The infrastructure of Gumbel [37] includes the two activation layers of convolutional and Gumbel softmax functions and the correlation of the residual. The generator's output, $\hat{v}$, is the discriminator's evaluation and transmits the extracted frequency, enabling the visualization of repetition compositions.

Using random Gaussian noise, the generator model creates synthetic data examples with a distribution resembling genuine sample data. The discriminator model is then used to determine whether or not the generated synthetic data are accurate. The discriminator model plays the game against the generator model, which creates samples to fool it as much as possible. The discriminator model tries to avoid being tricked as much as possible. In the end, discriminator D and generator G will be in Nash equilibrium. Alternate training is carried out for D and G if the goal function value or the designated number of cycles falls short of the threshold. Using an Adam optimizer, the gradient update maximizes the D loss value L. The data set imbalance problem is resolved, the data set is rebuilt, and data are generated for the generator G model.

The cWGAN model was validated against the dataset following improvement using an RCNN and a BiLSTM. Convolutional layers are stacked before the pooling layer in our RCNN network. The activation function ReLU is positioned in between the convolutional layers by stacking them. The activation function can better learn the spatial feature information of the complicated high-dimensional network traffic data of the DGAs thanks to the stacking of nonlinear functions, which boosts its nonlinear expressiveness. Because of the LSTM's efficiency, sequence features can only be extracted in a single direction, and the influence of the features that come before and after in the DGA network data is not fully considered. In this study, we employ the BiLSTM to extract features from long-distance dependence network traffic information in both positive and negative directions. The results are then produced by Gumbel softmax after DGA detection.

4.1.4. Gumbel Softmax

The distribution of Gumbel softmax has been discussed separately through the investigations of two groups [37,38], and its sequence authorizes persistent inaccurate examples out of a sequence of definite examples and allows that Z may be an unconditional factor through the possibilities of class. The misleading Gumbel-Max permits the following examples from Z:

$$y_i = \frac{e^{\frac{\log(\vartheta_i)+g_i}{\mu}}}{\sum_{j=1}^{k} e^{\frac{\log(\vartheta_j)+g_j}{\mu}}} \tag{12}$$

where $g_1, g_2, \ldots, g_k$ represent examples from Gumbel (0,1) and those relative through the softmax function that are persistent and varied [37], and these samples are built straightforwardly by the node-like stochastic model. The backpropagation algorithm may not generate if the node has not changed. In the context of the generation of vocabulary [39], it

is suggested to apply a word distribution and determine the illustration of weighted typical phrases along with alternately weighting by extremely credible words and several initial flexible categories and illustrating inherent information through secluded discriminator levels. Two dense layers make up the output layer. The number of domain names, n, is the input of the first dense layer into the second dense layer with n hidden neurons. Gumbel softmax activation is used for DGA family classification tasks, whereas the sigmoid activation function is used for detection tasks.

4.2. Word-Based Feature Extraction

4.2.1. Text Representation

DGAs created by character-based as well as wordlist-based approaches are entirely varied, as either type of word identification needs to be adjusted accordingly, and the approaches generally regard two varieties of features, the visualization of word2vec and the consequence features of n-grams. The standard schemes of our approaches are displayed in Figure 1, as are the expressive features of word2vec [18]. In addition to the elements of the wordlist, DGA has terms like English, and all selected word features are discussed in Table 3. Against these words, predicated through the DGA as well as benign phrases, segmented words encompass dimensional-based statistical word features involving word2vec [40], keras [34], glove [42], or NBoW distributions [42] that enhance the anti-detection abilities of word DGAs.

As a consequence of features like n-gram, the algorithm utilizes the previous examples to characterize the morpheme connection between DGA domains. Therefore, the result of the succession features of word DGAs [27] and the subsequent attempt is the extraction of features, and the extracted significant features such as word segmentation are first used to calculate the word-embedding features such as word2vec [41], and FastText [42].

In the feature fusion and classification step, we use two fully connected layers to fuse the four features and utilize activation functions such as Gumbel softmax to compute the word-based classification result. When the value is more significant than 0.5, the DNS is classified as an AGD, and when the value is less than 0.5, the domain name is classified as a benign domain name. As shown in Figure 5, the overall concepts of the detection method in DGA domain names with feature fusion are all word distributions.

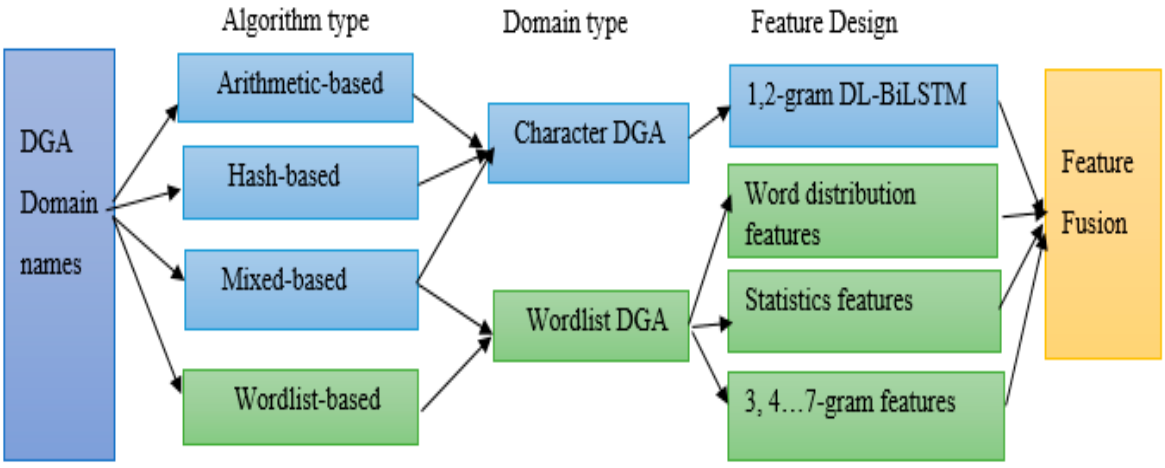


Figure 5. Overall concepts of the detection method.

4.2.2. Word Distribution Features

The last feature is the word reputation score (Word Reputation DGA). Here, we used the Tranco and DGA domains to generate a weight matrix to calculate the domain reputation value. The weight matrix begins by reading all the top domains from Tranco and DGA, then learning the vocabulary dictionary of n-grams of 3 to 7 characters or words,

and returning the term-document matrix. We used the base-10 logarithm function of the total *n*-gram [27] matrix of all Tranco domains, as shown in (4).

$$W_N(j) = log_{10}\left(\sum_{i=1}^{n} W_N(j)\right) \quad (13)$$

where *W* is a consequence matrix applied before the computation of the score of the word reputation, presented as the frequency of a word *n*-gram, and immediately misleading the score of victim domains' reputation. The initial excerpt scores taken out of the domain victim exploit the language *n*-gram [27] creator of the word, and internal computation is identical to creating the document-word matrix from Tranco [44]. The selected word representation features and description in Table 4 outline these word features to make a sequence of word-like features and the *n*-gram-based [43] consequences of the extraction of features predicated on the 3, 4, 5, 6, and 7-g sequences, respectively, as discussed.

Table 4. Selected word representation features and descriptions.

Word Representation	Description
Neural Bag-of-words (NBoW)	$NB = \frac{1}{X}\sum_{w \epsilon a} w_t$ (14) $Y1 = \text{softmax}(t_1 z + b)$ (15) where *a* indicates the input text, *t* indicates the word, and w_t indicates the word vectors.
FastText	$Z = -\frac{1}{N}\sum_{n=1}^{N} b_n \log(f(vua_n))$ (16) where *N* indicates the numeral of the document, a_n is the regularized bag of words identifying the *n*th document, b_n is the class label, *u* and *v* are the consequence matrices.
Word Repetition Rate	$WRR = \left(\prod_{m=1}^{4} \frac{\sum_s (x(m) - x(m,1))}{\sum_s x(m)}\right)^{1/4}$ (17) where *S* indicates the sliding window, *x*(*m*,1) indicates the integral of singleton *n*-gram [43] varieties in *S*, and *w*(*m*) is the unconditional integral of *n*-gram varieties in *S*.

4.2.3. Requirement of DGA Attack

Word Repetition Rate and Collision Rate: We recently used the word repetition rate to determine the route of repetition within the dictionary through the expression of the ratio of non-singletons like *n*-grams [27] ($n = 3, \ldots, 7$). WRR ranges from 0 to 1, and consequently, the moment is addressed to the perfect *n*-grams monitored in the exact interspaces of words (WRR is 0) and also formerly (WRR is 1). Then, analogous words are learned to expand the vocabulary, achieved by applying a sliding window to conserve significant feasible consecutive frameworks of the initial word. Consequently, highlights of the words to be processed together are sampled, and the cWGAN adversarial is applied for oversampling the domain name generation algorithm, and the training of the generator to optimize all of the generated examples is summarized in Algorithm 1. The inputs of Algorithm 1 are a list of domain sizes, random functions, word rule functions, segmentations, process generations, and target models. After labelling, data preprocessing is performed. The outputs require a set of adversaries with unrepeated word domains.

Collision Rate: The collision rate is applicable for domains conflicting within the catalogue and the domain created through DGA, and the result of the initial primitive success rate is chosen as the bot that combines with the C&C server.

Wordninja [40] word representation packages, in virtue, handle the distribution of every class label, and then the algorithm displays the extracted word-level category of the adversarial, which is summarized, and the changes to the word vector's unconditional phrases are shown by focusing on each phase and word sequence produced by the domain generation and target models, preventing the issue of repetition.

Algorithm 1: WordDGA

Input: Domain size m, random function $\Phi(.)$, word rule function R(.), segmentation function s(.), DGA domains $x = \{x^{(i)}, \ldots, x^{(m)}\}$, Generation model $\mathcal{G}(.)$, target model $\mathcal{M}(.)$, words $y = \{y^{(1)}, \ldots, y^{(n)}\}$,
Output: word unrepeated Domains $z = \{z^{(i)}, \ldots, z^{(m)}\}$
1: Begin
2: $\mathcal{X}_n \leftarrow \mathcal{X}[label = 0]$, $\mathcal{X}_b \leftarrow \mathcal{X}[label = 0]$;
3: Preprocess data;
4: for i in $1, \ldots, m$; do
5: for j in $1, \ldots, n$; do
6: $\mathcal{M}_{D_j} \leftarrow \mathcal{M}_{D_j} - \Phi_{D_j} \mathrm{s}(x_j)$
7: $\mathcal{M}_{\mathcal{G}_j} \leftarrow \mathcal{M}_{\mathcal{G}_j} - \Phi_{\mathcal{G}_j} \mathrm{s}(z_j)$
8: $\mathcal{M}_{\mathcal{G}_i} \leftarrow \mathcal{M}_{\mathcal{G}_j}$
9: $\hat{y}^{(i)} \leftarrow cWGAN(m^{(i)})$
10: $\mathcal{W} = \mathcal{M}(\Phi(y^n))$
11: $z^{(i)} \leftarrow R(\mathcal{W}, s(x^{(i)}))$
12: While $z_j^{(i)}$ *in z is DGA; do*
13: $\mathcal{W}' = R\Big(s(x^{(i)}),\ y(\Phi(n))\Big)\mathrm{P}(\mathcal{W}|z) = 0$;
14: $z_j^{(i)} \leftarrow \mathcal{W}'$;
15: end while
16: $z = z + z^{(i)}$;
17: end

5. Experimental Design

Through a series of analyses, the implementation of the suggested DNN is predicted through the DGA botnet with the cWGAN-based oversampling method on datasets of multiple real-world DGAs.

5.1. Evaluation Criteria

To identify the initial implementation realized through conditioning towards the initial, imbalanced data are used to fill absent handling class labels distinct from samples that exclusively have a strategy of oversampling and executes only similar enhanced oversampling in the initial and the baseline, connecting between identical typically operated techniques of oversampling. SMOTE [2] is a popular oversampling approach, and then second interpretations in SMOTE, such as adaptive synthetic oversampling (ADASYN) [4] and the popular summarization, are combined with the cWGAN method, and the approaches of oversampling are predicated by cWGAN.

Data Extraction and Processing

To improve training and prediction accuracy, the correlation of the different features was determined using a matrix of Pearson's correlation coefficients [42]. Features with values higher than 0.9 were eliminated. The dataset was analyzed for imbalance and skew by removing address-related features and utilizing the Python MinMaxScaler (version 3.5), a Scikit-Learn module component. This function rescaled all features to the range [0,1], improving training performance by compressing inliers. APT-related and regular flow classes were established, with 70% of the data used for training and 30% for testing.

5.2. Dataset Preparation

In the preliminary research, issues in datasets are separated into three sections, such as benign, DGA, and adversarial DGA families, which are characterized for success.

Benign Datasets: To conclude the dataset through the legit domain names selected from the Tranco directory to the highest extraordinary million current public online service domain names. In the list of Tranco [44], the hierarchy service predicts the accessible hierarchy out of the Alexa, Majestic, Cisco Umbrella, and Quant datasets, and helps close

the hierarchy by reliably handling consent toward the selection of current domains and experience through the measuring of the hierarchy by the record and shortage of malicious domains [44]. To randomly select 100,000 datapoints, a whitelist of reputation features was built, and the prevailing 900,000 data points were allocated to the training set to discover the DGA classification.

DGA Datasets: Our experiment samples were classified into different notable datasets that include Bader's [45], DGArchive [46], UMUDGA [47], OSINT [48], 360Netlab [49] and UTL_DGA22 [50] which are briefly described below and summarized in Table 5.

Table 5. Descriptions of datasets.

No.	Botnet Families	Number of Domain Names	No	Botnet Families	Number of Domain Names
1	Rovnix(m)	50,000	24	Qadars	2000
2	Dyre	1000	25	Simda	4000
3	China	1000	26	Pykspa_v2	799
4	Fobber_v1	298	27	Locky	1158
5	Tinynuke	32	28	Pykspa_v2_real	199
6	Gameover	4000	29	Shifu	2546
7	Murofet	4000	30	Matsnu(w)	50,000
8	Cryptolocker	1000	31	Proslikefan	100
9	Padcrypt	168	32	Tempedreve	195
10	Dicrypt	762	33	Vawtrak	827
11	Fobber_v2	299	34	Symmi	1200
12	Vidro	100	35	Suppobox_1(w)	20,000
13	Emotet	4000	36	Suppobox_2(w)	20,000
14	Tinba	4000	37	Suppobox_3(w)	12,000
15	Ranbyus	4000	38	Nymaim	480
16	Shiotob	4000	39	Nymaim2 (w)	5000
17	Pykspa_v1	4000	40	Conficker	495
18	Gozi_gpl(w)	50,000	41	Bigviktor(w)	9990
19	Gozi_luther(w)	50,000	42	Gspy(w)	100
20	Gozi_nasa(w)	20,000	43	Enviserv	500
21	Gozi_rfc4343(w)	50,000	44	Banjori(m)	50,000
22	Kfos(w)	5000	45	Volatilecedar(m)	10,000
23	Ngioweb(w)	5000	46	Pizd(w)	10,000

Bader's [45]: A comprehensive collection of 48 DGA botnet families available on GitHub thanks to Johannes Bader's publication of the entire dataset [45]. New botnet families are being added to this dataset, first published in 2018.

DGArchive [46]: Daniel Plohmann is the administrator of the Fraunhofer FKIE service, DGArchive. This site compiles and offers datasets concerning the 86 distinct DGA botnet families.

UMUDGA [47]: This is a standardized dataset synthesized and used for experiments in many works. It was created through the research group at the University of Murcia and released in 2020. The dataset concludes that, to date, 50 different categories of DGA families have been measured. Its groups as many as 50,000 examples to malicious domains considering any of these DGA families and some 500,000 domains to Tranco [44] and DGAs outside of malicious datasets.

OSINT [48]: This dataset is similar to the malicious dataset generated through Bambenek Consulting in that it contains a substantial quantity of private malicious DGAs composed and compiled. The scientific community intentionally shares this information, and it is not an authorized message exploit.

360NetLab [49]: These datasets compile public websites associated with the latest DGA malicious content, obtained through the analysis of data from current real-world DGAs such as Matsnu, Suppobox, Rovnix, and Banjori, constructed by the 360Netlab research group, Qihoo 360 security technology Company, Ltd., Beijing, China. This dataset merges malicious domains in both the DGA domain and in lower than DGA domains.

UTL_DGA22 [50]: With 4.3 million entries from 76 DGA families, UTL_DGA22 is a Domain Generation Algorithm dataset of botnet domain names released in 2022 [50]. With 76 DGA botnet families combined and presented, it can be regarded as the most up-to-date and comprehensive specialized dataset on the DGA botnet.

We used multiple domain name classification datasets to assess our model's performance. We employed Bader's, OSINT, DGArchive, 360Netlab, UMUDGA [47], and UTL_DGA22 [50] for the classification of harmful domain names and used the Tranco [44] list for benign domain names. To obtain 10,000, 20,000, or 50,000 domain names per family, we also enhanced domain data by generating script files.

We constructed six datasets: a wordlist of uncommonly collected DGA samples and a complete dataset of highly collected samples. The dataset is split into 70% of examples persisting before the dataset for the training set and the remaining 30% for the testing set. When the number of samples in each class increased, the accuracy of the training set decreased significantly. It registered the approaches to fulfilling the complicated tasks for the concluding issue.

The prompt makes the dataset predicated in Tranco [44], UTL_DGA22 [50], UMUDGA [47] and includes the legit domain names as well as those in the UMUDGA library, which involves 50 DGA families, containing 9 families with 13 various wordlist DGAs and 3 families with mixed-based DGAs, and the experimentation has recognized several for autonomous DGA families. In previously exploited DGArchive [46] and Netlab360 [49] datasets, the malicious domains are determined from actual traffic, and the composition of DGA examples for every family is widely varied. As a result, a rigorous extension is applied to the design involving three testing sets of that comprise the DGA families' datasets to attempt to identify awareness examples and three comparative approaches that are informed against the total dataset.

Adversarial DGA: After that, WordDGA will be identified according to the six modern adversarial DGAs: DeepDGA, DeceptionDGA, MaskDGA, Khaos, CLETer, and ReplaceDGA. Every DGA is trained to create a DNS, and specifically, Khaos and WordDGA randomly choose 100,000 examples from legitimate data considering the WordDGA. The development and the lasting domain before the benign dataset work with the addition of Khaos and ReplaceDGA to produce adversarial DGAs. DGA-based botnets work with the addition of all adversarial DGAs, except ReplaceDGA, created through casual vectors to produce benign domains and are developed through the six modern adversarial DGAs and WordDGAs in the set of training and testing DGA classifications to assess the capacities of WordDGA and diverse active adversarial DGAs.

5.3. Evaluation Metrics

$$\begin{cases} precision = \frac{TP}{TP+FP} \\ DR = \frac{\sum TP}{\sum TP+\sum FN} \\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \\ FPR = \frac{FP}{FP+TN} \\ F1\ score = \frac{2*Precision*DR}{Precision+DR} \\ FNR = \frac{FN}{FN+TP} = 1 - TPR \end{cases} \quad (18)$$

To execute the appraisal of WordDGA, five benchmark criteria are used with three datasets. The composition of positive examples labelled as DGAs and negative examples labelled as benign, especially based on the evaluation criteria in Equation (17), is labelled as *TP* (true positive) and represents the ratio of correctly classified DGA botnets to the total sample. *FP* is the ratio of legitimate examples severely mistyped, and *TN* represents the ratio of legitimate examples correctly classified. *FN* represents the ratio before the DGA domain was incorrectly classified. The *F*1-score is a comprehensive consideration of precision and *DR*; that is why the harmonic mean and improved *F*1 should be resolved.

These proposed DNS-based botnet attack identification models consider the information two synthetic or domain names. They also take a value of 0 for comparable and 1 for different, and data merging may be '0' for comparable and '1' for different. The merging of data may serve the GAN based on oversampling collected before combining the deep learning infrastructure and the word2vec level of embedding for the original layer. This diverts the recorded characteristic of the vocabulary of the vector of dense layer before a particular dimension. Considering the range of every combination, pre-processing is performed, and then the word [51] is created.

AUC-PR: The outlines of this angle allow for an analysis of the compromise between the precision rate and recall half edges. The proportion of TPs is the precision of completely categorized examples. and the classification performance is usually incredibly optimistic [52].

$$\text{AUC-PR} = \sum_n (rec_n - rec_{n-1}) \times prec_n \tag{19}$$

where rec_n and $prec_n$ are the recall and precision at the n-th edge.

5.4. Experimental Results and Observations

In this work, DNN techniques, especially the generative adversarial networks with oversampling, were used to distinguish whether DNSs were either similar or not, and various DNNs connected with RNN, CNN, and BiLSTM for identifying DNSs as either benign or DGAs are discussed in detail.

5.4.1. Experimental Setting

The entire series of DL experiments was carried out on a keras [34] GPU workstation embedded in an NVIDIA Geforce RTX running Ubuntu 20.04, with an Intel(R) Core (TM) i7-3230M CPU @ 3.00 GHz and operated by a Jupyter 7.2.2 notebook computing platform. We adopted keras, sklearn, and TensorFlow packages and adopted n-fold cross-validation, which is generally applied when distinguishing domains of training and testing sets. The learning rate was the Adam optimizer [53] r = 0.0002, and the batch size is 128.

The implementation is to test five modern DGA classification algorithms, namely BiLSTM, CNN-LSTM [54], CNN-BiLSTM-GAN, and RCNN-BiLSTM-cWGAN, in detecting DGA domains created through WordDGA; categories were separated into the two types of predicates in those algorithms.

To evaluate the identification capacities of these malicious DGA classifiers created through classical DGA algorithms, we combined domain datasets such as Tranco [44] with malicious DGA datasets, and we randomly designated 70% of the datasets before their combination to the training set and the residual 30% for the test set. The classification of five DGAs was implemented in the training set. In the testing set, the number of detected matches with existing DGAs was analyzed in advance of the criterion of competence for anti-detecting WordDGAs that mix benign and malicious DGAs produced through the WordDGA. Finally, the standard implementation before the five different classifications of DGAs determines the DNS created through WordDGA.

As a DNN mechanism based on oversampling methods, the hyperparameter rate play a vital role in the achievements of WordDGA. Various series before the training and testing to WordDGA have informed the use of distinguishable values to the rate of hyperparameter.

The backend libraries of TensorFlow and Keras were used to implement the modeling experiment in Python 3.9 on the Anaconda JupyterLab platform. The models underwent

5000 epochs of training, with an average learning rate of 2 $\times 10^{-4}$, and were optimized using the optimizer developed by Adam [53]. Utilizing a 3.00 GHz Core i7 CPU (8 cores) with 32 GB RAM, training cGAN took nearly an hour and forty minutes, while training WGAN and cWGAN took about three hours. While D's structure consisted of a layer for the input, five hidden layers, and a ReLU (rectified linear unit) activation for every layer except for the output layer, which featured a sigmoid activation function, G's structure consisted of five layers containing three hidden layers. However, C did not employ the sigmoid activation function at the output. The authors of [55] discuss standard activation functions. For neural network training, a piecewise continuous activation function such as ReLU is often employed because it solves the vanishing gradient issue, which stops gradient information from being backpropagated through the network. A non-linear activation function that brings values into the range [0,1] is a sigmoid or tanh activation function utilized in models for output probability prediction. For the reader's convenience, we have included a summary of the GAN [56] parameters in Table 6.

Table 6. Training parameters for the applied GAN models.

GAN Parameter	cGAN	WGAN	cWGAN
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}
Optimization	Adam	Adam	Adam
Loss	Minimax	Wasserstein	Wasserstein
Epochs	5000	5000	5000
Batch size	128	128	128
Layer activation	LeakyReLU	LeakyReLU	LeakyReLU
G o/p activation	Tanh	Tanh	Tanh
D o/p activation	sigmoid	-	-
Dropout	0.2	0.2	0.2

The list of the top one million domain names that Tranco gathered includes benign domain names. To create the benign domain dataset, 950,000 domain names were chosen from Tranco. The dataset collected contained the DGA domain name samples. The 56 types of DGA botnet families were selected, and 30,000 samples were taken for each type.

The data sources yield a total of 1,900,000 domain names. Table 7 displays the dataset description and distribution. Other domain names that are more challenging to identify as DGA domains are those that contain common and readable English words (wordlist-based DGAs [57]), and the training set makes up 80% of the dataset, whereas the testing set makes up 20%.

A comparison between the proposed model and three deep learning models is planned. The results of the proposed and comparative models' evaluations on the DGA detection task are displayed in Table 6.

Table 7. Benign and DGA domain name dataset description (DN—Domain names).

Dataset	Type	Year	Task	Num Classes	Size	Training	Testing
Tranco	Benign	2019	Normal	-	1,000,000	999,858	73,694
UMUDGA	DN	2020	DGA	51	3,098,626	1,853,175	616,726
UTL_DGA22	DN	2022	DGA	75	4,297,916	2,478,745	849,574
Bader's	DN	2018	DGA	48	Null	31,457	12,975
DGArchive	DN	2020	DGA	86	Null	200,925	99,784
OSINT	DN	2021	DGA	-	495,186	357,274	130,573
360NetLab	DN	2021	DGA	-	Null	500,000	249,348

5.4.2. Comparison with Existent Adversarial DGAs

These sections analyze and discuss the characteristics of DGAs produced through WordDGA as well as compare the anti-detection performance of WordDGA against various adversarial attacks. The progress of the domain retention of the newly enhanced modifier that contains DeepDGA [6], Khaos [7], DeceptionDGA [8], MaskDGA [9], CLETer [10], and replaceDGA [11] is compared, and attempts of adversarial attacks, especially evasion attacks, are utilized in advance of the evasion attack and deep learning techniques [58].

To execute all adversarial-based malicious attacks better than the results of existent adversarial networks with and without training and testing, the RCNN-cWGAN-BiLSTM was retrained on the recently created DGAs from MaskDGA and CLETer. The results in Table 8 show that training is improved through executions.

Table 8. Performance of the proposed classifier compared to all the existent adversarial DGAs and WordDGAs without adversarial training.

Model	**RCNN-cWGAN-BiLSTM**						
Metrics	**DeepDGA**	**Khaos**	**MaskDGA**	**CLETer**	**DeceptionDGA**	**ReplaceDGA**	**WordDGA**
Precision	0.9916	0.9953	0.9979	0.9973	0.9887	0.7188	0.7098
Recall	0.4182	0.2426	0.5347	0.4561	0.1124	0.0541	0.0529
F1	0.5170	0.3900	0.6962	0.5968	0.1867	0.1101	0.4009
Accuracy	0.6943	0.6207	0.7668	0.7534	0.5549	0.5165	0.5059

In these seven existent and current adversarial-based DGAs, we compare the implementation performance before DGA-based botnet identification with and without adversarial training technology, which refers to adding the domains produced through the corresponding adversarial-based training set to train the DGA classifier. Table 8 shows that all the evaluation criteria before the DNN classification of the existent adversarial sets of testing increase, finding the domains created through the WordDGA to be substantially smaller compared to the different adversaries. According to Table 9, the evaluation metrics of these DNN classifications have improved through the adversarial training. All criteria before DNN-based DGA detection created through DeceptionDGA and Khaos have been enhanced to above 0.8000, and the accuracy of DNN-based DGA domain detection produced through ReplaceDGA has been improved. The experimental results suggest that this presumed WordDGA quietly outperforms these seven different current methods of detecting adversaries despite this adversarial training technology.

5.4.3. Performance Analysis with Previous Detection Methodologies

Performance Analysis with Effects of the Proposed Detection Method

The methodologies that we implemented were detection strategies [59] based on meaningful wordlist botnets. These RCNN-cWGAN-BiLSTM techniques predicted affiliate highlights removed through RCNN, and the BiLSTM approach [19] predicted character- as well as word-based DGAs. For RCNN-cWGAN-BiLSTM specified the original reference to BiLSTM and re-implemented these techniques based on word2vec as per these explanations by these works, and the performances of the four approaches are displayed in Table 3. In this subsection, we analyze the evaluation metrics of the oversampling techniques, such as SMOTE and ADASYN models, to assess their performance on classification-based DNN [60] traffic examples in the test set shown in Table 10.

Table 9. Performance of the proposed classifier compared to all the existent adversarial DGAs and WordDGAs with adversarial training.

Model	**RCNN-cWGAN-BiLSTM**						
Metrics	**DeepDGA**	**Khaos**	**MaskDGA**	**CLETer**	**DeceptionDGA**	**ReplaceDGA**	**WordDGA**
Precision	0.8398	0.9577	0.9634	0.9510	0.8053	0.7128	0.7122
Recall	0.9367	0.8358	0.9942	0.9540	0.8323	0.6126	0.6018
F1	0.9267	0.8926	0.9751	0.9524	0.8186	0.6532	0.6415
Accuracy	0.9258	0.8994	0.9782	0.9524	0.8156	0.6753	0.6154

Table 10. Classification experiments of correlation performance in DL models all metrics.

Dataset	**Indicators**	**Oversampling**	**Ours**	**CNN-LSTM**	**CNN-BiLSTM**	**BiLSTM**
Dictionary-based dataset	Accuracy	None	96.33	94.39	94.13	93.09
		SMOTE	98.31	98.22	98.23	92.89
		ADASYN	99.93	98.46	98.59	92.62
	Precision	None	96.36	94.35	93.61	92.89
		SMOTE	98.78	97.66	98.19	96.31
		ADASYN	99.18	98.29	98.92	97.36
	Recall	None	96.65	94.96	95.25	93.97
		SMOTE	97.56	96.45	97.26	96.01
		ADASYN	98.56	97.16	98.37	96.08
	F1	None	96.49	94.65	94.43	93.42
		SMOTE	96.58	95.23	96.12	94.56
		ADASYN	97.22	96.89	97.01	95.68
Adversarial dataset	Accuracy	None	98.64	96.79	98.14	98.42
		SMOTE	99.18	98.89	99.06	97.56
		ADASYN	99.80	98.59	99.31	97.82
	Precision	None	98.69	96.73	98.54	98.59
		SMOTE	97.32	96.67	96.99	95.63
		ADASYN	99.68	98.78	99.26	97.49
	Recall	None	98.59	96.79	97.69	98.24
		SMOTE	98.87	97.36	98.54	96.96
		ADASYN	99.17	98.59	98.99	97.16
	F1	None	98.63	96.79	98.15	98.41
		SMOTE	98.79	97.59	98.21	97.08
		ADASYN	98.92	97.64	98.16	97.13
Full dataset	Accuracy	None	97.29	94.64	96.15	96.13
		SMOTE	97.69	96.56	97.12	96.23
		ADASYN	98.89	97.34	98.32	96.43
	Precision	None	97.22	94.79	96.19	96.73
		SMOTE	97.89	96.62	97.19	96.08
		ADASYN	98.58	97.26	98.07	96.99

Table 10. *Cont.*

Dataset	Indicators	Oversampling	Ours	CNN-LSTM	CNN-BiLSTM	BiLSTM
Full dataset	Recall	None	97.25	94.47	96.13	95.49
		SMOTE	97.99	96.54	97.21	96.19
		ADASYN	98.57	97.76	98.01	97.03
	F1	None	97.24	94.63	96.16	96.11
		SMOTE	97.76	96.23	97.26	95.99
		ADASYN	98.43	97.24	97.99	96.87

In the context of the category-based distinctiveness, through the training set, examples of SMOTE accuracy and F1 have increased as long as they are all in the same dataset. Similarly, evaluation metrics before the addition of ADASYN as these prevailing families and still these evaluation metrics of the form of ADASYN as these levels of a minority have moderately less connection to these models for these datasets.

With this combination of efficiency before these features, several techniques substantially developed this execution in the dataset of total and wordlist. When these methods were applied to wordlist DGAs for highly similar classes, like Matsnu and Nymaim2, the AUC-precision was somewhat raised to 94.9% and 96.2%, respectively. In addition, these moderate classes of AUC-precision before the 14 wordlist classes perform terribly before mixed-DGAs was added to these classes, and the ratio of AUC-precision before wordlist-based DGAs marginally increased.

In mixed-DGA classes of the algorithm, the edges and the variants of the proposed Suppobox display as edges commanding through the model of detection rate value, such as 0.994, and the values of different techniques are between 0.975 and 0.980. All approaches to detecting 14 wordlist-based DGAs are shown in Table 11. Notoriously, the integers before the examples of Gozi and Suppobox are slight, and the classes of DGA domains have been collected to connect the phrases.

Table 11. Performances of detection models in all evaluation metrics in detecting 14 wordlist-based DGA botnets utilizing the complete datasets; the best results are marked in bold.

Domain Type	AUC-Precision			Detection Rate			F1		
	Proposed	CNN-BiLSTM	BiLSTM	Proposed	CNN-BiLSTM	BiLSTM	Proposed	CNN-BiLSTM	BiLSTM
Benign	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Banjori	1.000	1.000	1.000	0.989	1.000	1.000	1.000	1.000	1.000
Bigviktor	0.995	0.982	0.964	0.978	0.946	0.934	0.994	0.854	0.772
Kfos	0.992	0.985	0.961	0.962	0.954	0.947	0.991	0.882	0.764
Ngioweb	0.991	0.981	0.959	0.969	0.952	0.944	0.989	0.889	0.769
Gozi_gpl	0.985	0.963	0.954	0.982	0.963	0.968	0.984	0.863	0.761
Gozi_luther	0.973	0.945	0.943	0.971	0.948	0.936	0.972	0.847	0.737
Gozi_nasa	0.946	0.889	0.883	0.939	0.919	0.897	0.939	0.809	0.791
Gozi_rfc4343	0.921	0.887	0.849	0.938	0.867	0.897	0.928	0.879	0.759
Matsnu	0.962	0.927	0.931	0.987	0.969	0.961	0.969	0.851	0.749
Nymaim2	0.949	0.956	0.928	0.935	0.935	0.926	0.939	0.843	0.735
Pizd	0.956	0.970	0.979	0.946	0.985	0.976	0.921	0.879	0.774

Table 11. *Cont.*

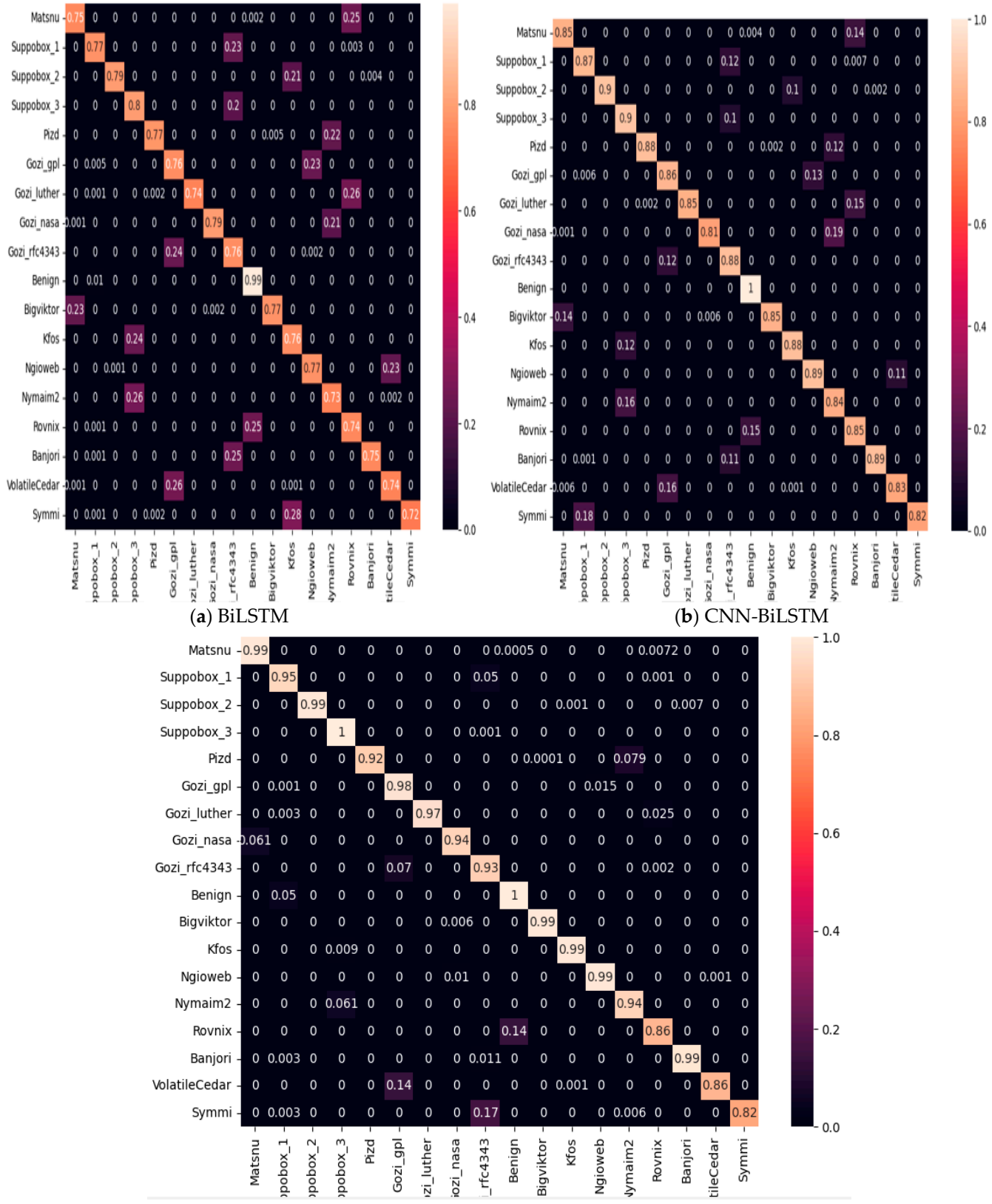
Domain Type	AUC-Precision			Detection Rate			F1		
	Proposed	CNN-BiLSTM	BiLSTM	Proposed	CNN-BiLSTM	BiLSTM	Proposed	CNN-BiLSTM	BiLSTM
Rovnix	0.869	0.852	0.821	0.857	0.849	0.834	0.862	0.853	0.744
Suppobox_1	0.929	0.966	0.960	0.974	0.975	0.980	0.931	0.871	0.769
Suppobox_2	0.992	0.982	0.984	0.994	0.996	0.984	0.992	0.890	0.789
Suppobox_3	0.999	0.995	0.996	0.999	0.999	0.991	0.999	0.897	0.795
Symmi	0.829	0.759	0.728	0.824	0.829	0.821	0.821	0.819	0.716
VolatileCedar	0.897	0.779	0.726	0.831	0.859	0.867	0.896	0.887	0.782
Khaos	0.889	0.873	0.829	0.846	0.793	0.735	0.875	0.826	0.762
MaskDGA	0.891	0.856	0.752	0.892	0.837	0.796	0.890	0.837	0.773
WordDGA	0.983	0.921	0.904	0.995	0.936	0.899	0.958	0.848	0.786

In all other DNN algorithms, the detection rate against the banjori and volatile cedar families reached 0.99 or zero based on a mixed DGA dataset of both characters and words. This is essential because the length of this domain is longer, and it is not quite the same as the length distribution of the benign category and the extension of a massive number of examples, like banjori. The detection rate of the proposed technique reaches 0.9178, and the values of the other three methods are somewhere in the range of 0.3 and 0.73. In wordlists, DGAs such as Suppobox combine arbitrary words before two subwords of DGA samples.

The minimum number of DGAs has been set according to Bigviktor, with a detection rate of 76.4% and precision of 91% traced through the Khaos, Suppobox, Gozi, and Matsnu by F1, classifying them at 95–98%. The Gozi variant has a low detection rate for wordlist-based botnets. The following table shows that the model has a 0.993 accuracy rate, which is relatively high for detecting the DGA botnet's domain name. AUC-precision and recall values that are highly similar between the labels demonstrate that the capacity to recognize labels is also uniform, not diverging from either side. In the Confusion matrix in Figure 6, additional insight into the model's capacity for classification is also displayed when the model is being evaluated. The reality of word-DGA classes to improve that predicated probabilistic methodologies[12] was exciting but challenging to the upcoming word exploration. The adversarial [61] plan these AUC-Pre acquired is dense with widespread wordlist DGAs, displaying the inconvenience of securing similar classes. Propagation with a measure of F1 is additionally displayed in boxplots in Figure 7.

In this study, word embedding, fastText, and TF-IDF represent the features, and *n*-grams separate the input data, consisting of domain names—benign and malicious—labeled appropriately. According to the results, when tested on the new dataset, the suggested algorithms continue to achieve high accuracy.

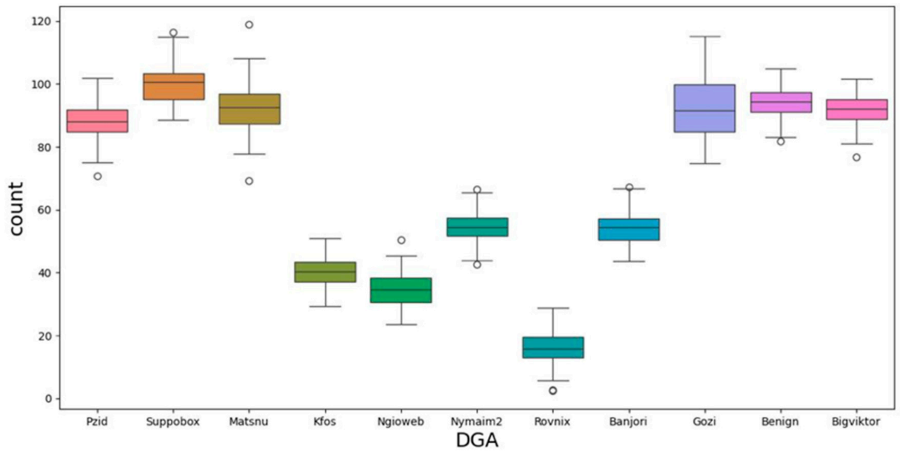
This work aims to identify word-based DGA botnet groups and harmful domain names. The overall findings demonstrate that the suggested model performs well across all datasets for most evaluation metrics. The proposed model performs better than the other cutting-edge model's domain generation method classification datasets on most assessment metrics. In the UTL_DGA22 [50] dataset, however, the CNN-BiLSTM obtains better results in F1 score and AUC-precision.



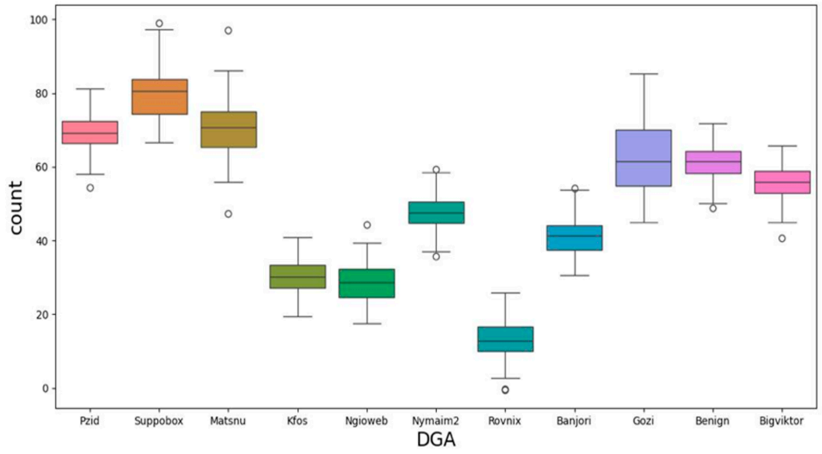
(**a**) BiLSTM (**b**) CNN-BiLSTM

(**c**) Proposed

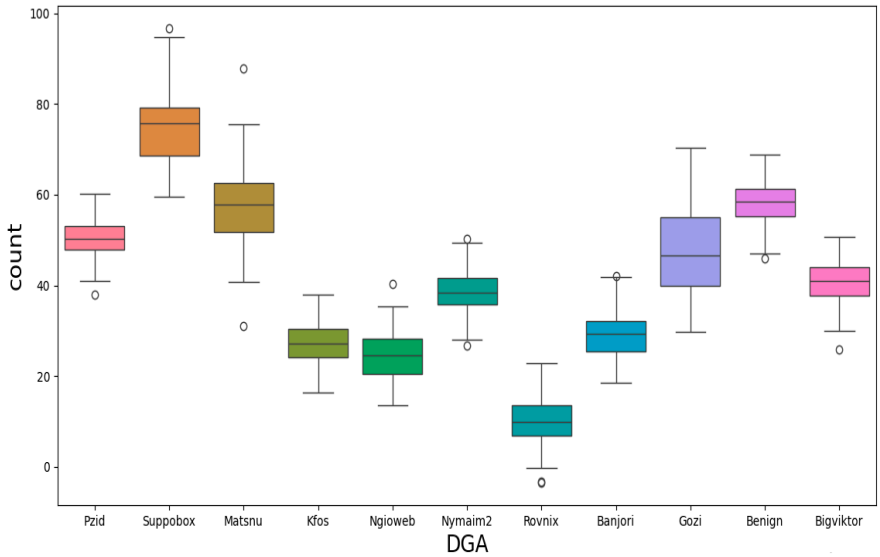
Figure 6. Confusion matrices of proposed DNN model-based word-based botnet with detection ability: (**a**) BiLSTM, (**b**) CNN-BiLSTM, (**c**) proposed model.



(**a**) Proposed



(**b**) CNN-BiLSTM



(**c**) BiLSTM

Figure 7. Boxplots of the F1 scores for the word list-based DGAs. The proposed model performs best compared with the previous techniques regarding character and wordlist DGAs botnets using word tokenization.

The results demonstrate the model's better accuracy of 0.98 on the newly released UTL_DGA22 dataset. With an F1-score of 0.97, thirteen families of word-based DGA botnets have been found with nearly perfect labels. With significant levels of similarities and good AUC-precision and recall, the remaining DGA botnet families typically produce good classification results.

The word DGA of Suppobox may be created through the DGA model for typical vocabulary phrases such as coriandertimothyson.net and the arbitrary substitute of these letters that desire to entirely destroy the lexicographical model of Suppobox DGA. Thus, the arbitrary attempt has been efficient as a variant of Suppobox. When WordDGA is applied to the bigviktor and kfos wordlist, targeted DGA examples precisely identify DGA-based adversaries by the classes of F1 of 0.586 and a moderate AUC-Pre of 0.829. The exploration of widespread classes displayed in Table 3 uses this analogous identity value on the comparable datasets as significant and complicated previous to the preferred average identification value that concludes secure them identically.

5.4.4. Analysis of Word Repetition Rate, Collision Rate, and Attack Success Rate

To analyze the three characteristics of the DGAs created through applying all criteria to be considered and related to attempts made in Section 5.3, the DGA must reserve these sequential features as necessities before its actual usage for exploration:

Word Repetition Rate: It reveals the classes of repetition of seven different DGAs and exposes all rates before seven existing DGAs in Table 11. In the context of actual DGAs, the values of repetition before Matsnu and Suppobox have significantly improved and have an enhanced value of repetition of 0.7 and smaller than 0.466, as demonstrated in Table 9. By exploring DGAs, the repetition rates for MaskDGA, DeepDGA, and CLETer have improved to 0.2 and trouble far before Khaos has exclusively better results.

The collision rate is not likely to collide with the appropriate domain, which has been catalogued for exhibiting independent and different collision rates, which are relatively lower, which signifies these conclusions accurately produce the recorded DGAs. This proposed model performs best in comparison with the existing techniques through character and wordlist DGA botnets using word tokenization, as shown in Figure 7.

So, using recently identified footsteps before the word DGA by examples of complicated ones like MaskDGA, Khaos, and DeepDGA, and wordlist DGAs such as Suppobox, Gozi, Matsnu, and Khaos, shown in Table 12, allow precise real-world-based word DGAs.

Table 12. Rates of all word repetition, collision, and attack success before the families.

Type	DGA	Word Repetition Rate	Collision Rate	Attack Success Rate
Research DGA	Khaos	0.64373	0.03290	0.648
	DeepDGA	0.37795	0.13361	0.524
	MaskDGA	0.71664	0.45948	0.873
	DeceptionDGA	0.85373	0.07605	0.765
	CLETer	0.56981	0.48269	0.895
	ReplaceDGA	0.02134	0.16584	0.819
	WordDGA	0.02036	0.11268	0.899
Real-life DGA	Matsnu	0.09992	0.00785	0.935
	Suppobox_1	0.01369	0.01560	0.929
	Suppobox_2	0.08291	0.01286	0.939
	Suppobox_3	0.70875	0.00564	0.934
	Gozi_gpl	0.01160	0.00013	0.886
	Gozi_luther	0.61109	0.01358	0.891
	Gozi_nasa	0.04186	0.11548	0.894
	Pizd	0.09876	0.08962	0.915

Table 12. *Cont.*

Type	DGA	Word Repetition Rate	Collision Rate	Attack Success Rate
Real-life DGA	Bigviktor	0.08264	0.07258	0.938
	Kfos	0.08467	0.03987	0.938
	Nymaim2	0.46662	0.05698	0.999
	Ngioweb	0.57369	0.11897	0.942

Suppobox, Pizd, and Matsnu are word-based, where a DGA domain is created by linking a grouping of word reference entries, and the extensions for vocabularies are slightly inclined to produce indistinguishable DGAs along with better results.

As shown in Figure 8, the standard of all rates before replaceDGA received 0.02%, as did the minimum amount of all rates between seven existent adversarial DGAs. The rate of collision was 0.165% and generally improved significantly before DeepDGA. It is accepted that the probability of all character-based sequences of DeceptionDGA and GAN-predicated DeepDGA executes moderately incorrectly with all rates. Still, WordDGA based on cWGAN performed better than previous research regarding repetition, collision, and attack success rates. Wordlist DGAs such as Gozi, Matsnu, Pizd, Bigviktor, and Suppobox identically tolerate the increase in all rates considering phrases through vocabulary have been repeatedly exploited, and Gozi wordlist DGA showed inferior rates of word repetition than other wordlist DGAs and has combined both characters and phrases.

5.4.5. Comparability in Research of WordDGA

The distribution of WordDGA research has more analyses of the implementation of the identified WordDGA by CLETer and MaskDGA. The two preferred exploration DGAs comprise the subsequent three phrases, such as the awareness of ASR below the proposed based DGA, like the rate of word repetition, collision, and the correlation of outcomes displayed in Figure 8.

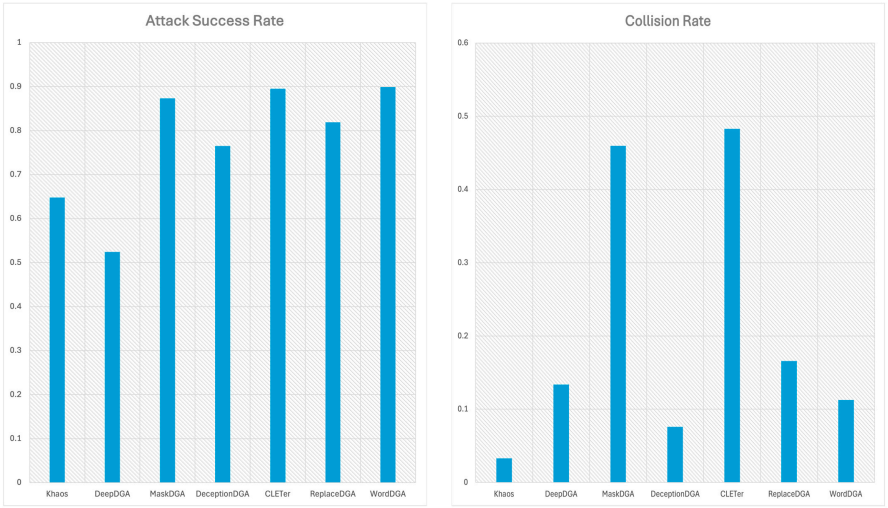
WordDGA shows the best detection competence. The work of Khaos and MaskDGA has average competence of detection by the ASR before 0.65 and 0.91, respectively, with the DeceptionDGA and Khaos, which are analogous positions of controlled competence of detection by the ASR rates discretely. The plots compare the different character- and wordlist-based DGAs for detection approaches in Figure 9 also showing benign ones.

Although ReplaceDGA performed well, it has a non-lightweight repetition rate and is significant in advance of the different DGAs. In Khaos and DeceptionDGA, the lightweight rate of collision is transferred, except that distinguishable patterns perform it, and that increases similarly to the checklist roughly created through the exploration of DGAs and domains out of these benign Tranco DGAs [44], as shown in Table 6.

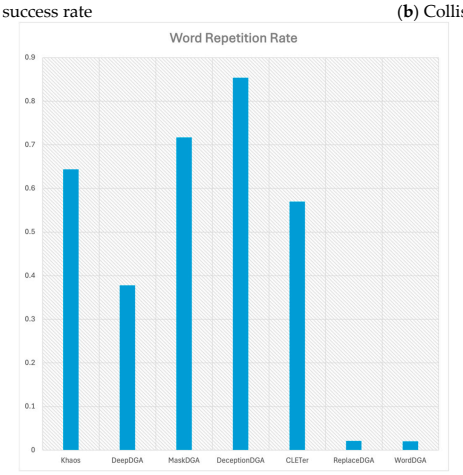
The model obtains an accuracy of 98.32% on the UMUDGA dataset, 99.39% on the 360NetLab dataset [49], and 97.73% on the UTL_DGA22 dataset, according to the assessments for the binary classification issue on each dataset. These outcomes came from using the same dataset for both training and evaluation.

5.5. Ablation Experiments for Effectiveness of n-Gram Consequences

In general, among *n*-grams [27], the consequence of techniques primarily involves 3, . . . , 7-g techniques that utilize the standard algorithms of *n*-grams to acquire their results and utilize these BiLSTM for sequence word feature extraction as well as classifiers. Here, we compare the results before the detection of every discrete *n*-gram consequence together with the results of all criteria of component combinations of strategies utilizing 3,4,5,6,7-gram that nominate the consequence features of *n*-grams, and the preliminary outcomes are displayed in Table 13.

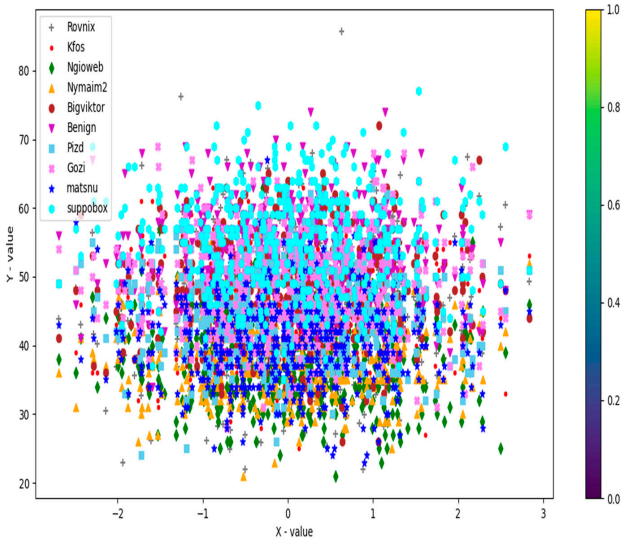


(**a**) Attack success rate (**b**) Collision rate



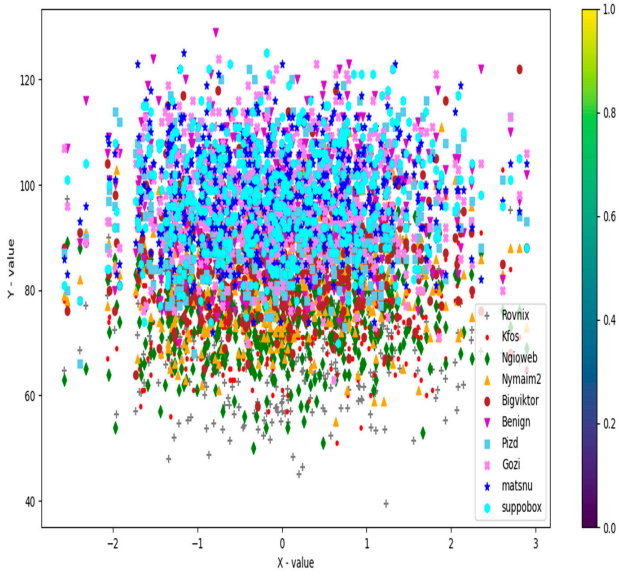
(**c**) Word repetition rate

Figure 8. Comparability of all rates in existent adversarial networks: (**a**) attack success rates compared to RCNN-BiLSTM model, (**b**) rates of collision in each adversarial, and (**c**) word repetition in each adversarial.

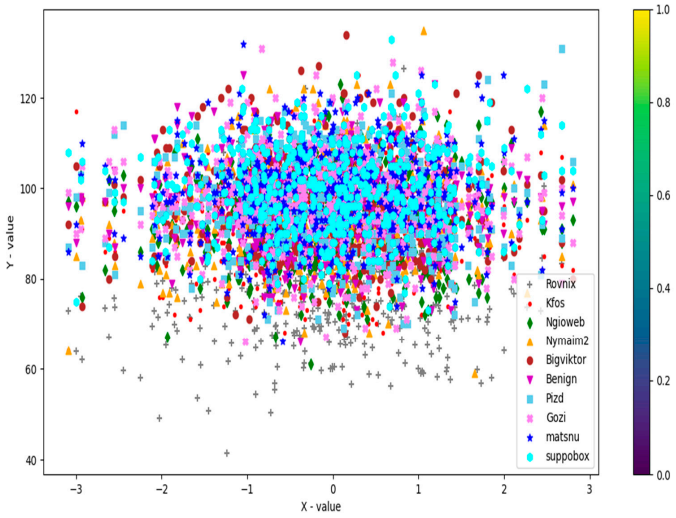


(**a**) BiLSTM

Figure 9. *Cont.*



(**b**) CNN-BiLSTM



(**c**) Proposed model

Figure 9. Comparison plots with different character- and wordlist-based DGAs for detection approaches.

Table 13. Execution on various consequence features of *n*-gram terms with all criteria and false negative rate (FNR).

	3,4,5-g					5,6,7-g				
Family	**Acc**	**Prec**	**Recall**	**F1**	**FNR**	**Acc**	**Prec**	**Recall**	**F1**	**FNR**
Benign	0.9999	1.000	0.9999	1.000	0.0001	1.000	1.000	1.000	1.000	0.000
Banjori	0.9998	0.9997	0.9998	0.9999	0.0002	0.9999	0.9998	0.9999	0.9998	0.0001
Bigviktor	0.9991	0.9989	0.9990	0.9989	0.0009	0.9995	0.9991	0.9989	0.9991	0.0005
Kfos	0.9989	0.9982	0.9979	0.9988	0.0011	0.9891	0.9982	0.9979	0.9989	0.0009
Gozi_gpl	0.9665	0.9251	0.8770	0.8659	0.0335	0.9855	0.9251	0.8770	0.8659	0.0145

Table 13. *Cont.*

	3,4,5-g					5,6,7-g				
Family	**Acc**	**Prec**	**Recall**	**F1**	**FNR**	**Acc**	**Prec**	**Recall**	**F1**	**FNR**
Gozi_luther	0.9534	0.8321	0.8996	0.8564	0.0466	0.9845	0.8321	0.8996	0.8564	0.0155
Gozi_nasa	0.9611	0.8836	0.8987	0.8756	0.0389	0.9858	0.8836	0.8987	0.8756	0.0142
Gozi_rfc4343	0.9678	0.8878	0.8857	0.8597	0.0322	0.9878	0.8878	0.8857	0.8597	0.0122
Matsnu	0.9923	0.8256	0.8498	0.7999	0.0077	0.9923	0.8256	0.8498	0.7999	0.0077
Ngioweb	0.9947	0.9886	0.8969	0.8978	0.0053	0.9865	0.9956	0.9899	0.9934	0.0035
Nymaim2	0.8772	0.5278	0.5866	0.5584	0.1228	0.9952	0.9278	0.9866	0.9584	0.0048
Pizd	0.9908	0.9784	0.9798	0.9544	0.0092	0.9979	0.9784	0.9798	0.9544	0.0021
Rovnix	0.9716	0.9788	0.9536	0.9301	0.0284	0.9716	0.9788	0.9536	0.9301	0.0284
Suppobox_1	0.9867	0.9821	0.9749	0.9546	0.0133	0.9917	0.9821	0.9749	0.9546	0.0083
Suppobox_2	0.9868	0.9562	0.9423	0.9479	0.0132	0.9968	0.9762	0.9623	0.9779	0.0032
Suppobox_3	0.9985	0.9813	0.9958	0.9958	0.0015	0.9995	0.9813	0.9958	0.9958	0.0005

As a consequence of a valid awareness of achievement considering the legal tender DGA varieties along with the identical experience through the shared categories of all *n*-grams with this criterion, since only 5, 6, and 7-g are consequences of attributes with the technique, they notably persist. These wordlist DGAs, together with the enhancement of the different wordlist features, have a slight moderate effect, and the exhibition of these strategies are just approximately as exclusive as the BiLSTM technique.

5.6. Model Comparison Experiment

To obtain better results for training time, the SVM, XGBoost, CNN, RNN, Bi-LSTM, and the integrated detection models presented in this study were chosen for comparison with machine learning and deep learning techniques. Table 14 displays the experimental outcomes of training performance.

Table 14. Model training algorithm comparison.

Model	**Training Time (s)**
SVM	1424.7512
XGB	1131.4862
BiLSTM	2375.1865
CNN-BiLSTM	5334.6873
Proposed model with *n*-gram	6821.5721

The following parameters are used for the experiments: the network's remaining parameters are set to their default values; the RCNN-cWGAN hidden layer had an output dimension of 64; the ReLU activation function had an output dimension of 32; the BiLSTM hidden layer had an output dimension of 64; and the dropout layer's loss rate was 0.1.

5.7. Comparative Qualitative Analysis

Table 15 presents a qualitative comparison of the proposed method with six previously reported approaches. Behavior-based detection [29–31], ML-based detection [4], DL-based detection [5,34], approximated dictionary-based detection [36], black-list-based detection [18], DPI-based detection [23], and ML-based detection are some of these techniques. In this qualitative comparison, we focus on the five elements listed below: the effectiveness of DGA malware detection, the effectiveness of dictionary DGA malware detection, real-time malware detection, resilience to encryption, and network size dependence. We examine these methods' operational limits and detection accuracies using the

practical concerns of each approach discussed in Sections 2 and 5.3. The final two elements and detection performance comprise the operating restrictions.

Table 15. A qualitative comparison of our work with other widely used detection approaches.

	DGA Malware Detection	Wordlist DGA Detection	Real-Time Detection	Robust to Encryption	Network Scale Independent
Wang et al. [57]	✔	✔		✔	
Liu et al. [10]	✔			✔	✔
Anderson et al. [6]	✔		✔	✔	✔
Qiao et al. [24]	✔		✔	✔	✔
Hoang and Hong et al. [15]	✔	✔		✔	✔
Vranken et al. [32]	✔	✔		✔	✔
Our method	✔	✔		✔	✔

✔-Indicates the methodology is addressed in the given article.

This method focuses only on those domains for whom the name resolution satisfies the forward lookup request and on the NXDOMAIN message answer to minimize the processing overhead. Setting these two requirements reduces the overall number of domains that need to be determined. If an NXDOMAIN response is received, the domain is not registered. Consequently, there can be no data transmission related to name resolution. Because of this, our method does not need rigorous real-time identification. Therefore, a basic optimization should be enough to produce satisfactory results for real-world use.

6. Result Discussions and Analysis

In summary, our results were closer in most of the datasets used before the all-evaluation criteria on both variants of GAN and the conceptual RCNN-BiLSTM deep learning architectures. WordDGA generates adversarial DGA by replacing the benign and DGA-based input words. Specifically, we can look through the legitimate dataset to check if there is a comparable one. The given identifier simply determines domains that are identical or unique. Tranco is created through meaningful phrases of malicious DGAs such as WordDGA and covertly generates prominent false positives. We suggest creating new FastText feature classes based on domain families and 36 base attributes based on domains using TF-IDF, depending on the new dataset. Simultaneously, all suggested algorithms for the DGA botnet problem have the same evaluation criterion. A comprehensive approach to data gathering, preprocessing, and domain training was provided. Our findings show that the suggested model performs at the cutting edge on several datasets.

However, fine-tuning a domain-generic, pre-trained word-embedding strategy to the DGA domain on several comparison datasets exceeds the initial character-based and word-based models. The evaluation's findings demonstrate the great accuracy of both suggested models, particularly regarding the DGA detection of botnet issues. The new model has a better execution speed and dramatically increases accuracy compared to deep neural network algorithms.

Additionally, the condition of existence, together with the proposed infrastructure, may not identify the DGA-based word created out of three classes of word-based DGAs. In contrast, these three-word DGA classes have been found in word phrase recite domains and may not efficiently infrequent out of the benign like Tranco [44], and the probability of this well-known DGA created out of pizd, Matsnu, Bigviktor, Kfos, Ngioweb, Nymaim2, a variant of suppobox and Gozi may be distinguishably identified the extension to the facts of environments and execution of unproductive have realized whereas word based categories such as mastnu, Gozi variant, Pizd, Bigviktor, Suppobox variant with the either classification. The predominant may be because categories of DGA have predicted meaningful word-pronounceable DGA, and these four categories of DGA may not indeed be

infrequent out of these Tranco (Benign) [44]. For wordlist-based DGA botnets, both BiLSTM and CNN-BiLSTM do not perform well (0.851 and 0.749 F1 scores), and the proposed model shows better results (0.969).

The prohibition on these theoretical models may not be applicable in advance of ascertaining mixed DGAs like banjori and volatile cedar, and feature models may not find these DGAs. Nevertheless, as this workout of two-component models should be possibly disconnected, it does not have an overall impact on identifying theoretical role models. Nonetheless, character domain models are rarely competent at ascertaining wordlist DGAs and use some word distribution highlights to prepare a DNS domain dataset to distinguish word-based DGA families. However, word-based domain name classifiers may rarely experience or even have the ability to find a specific character DGA.

7. Conclusions and Future Work

This work proposes WordDGA, a different variant of adversarial DGA based on RCNN-BiLSTM with cWGAN, to recognize DNS for word-based DGAs on highlights of phrases after phrase embedding. The consequence of developing extra word features that produce domains has aided in detecting DGAs along with the anti-detection ability under the statistical recognition approaches concerning a variety of natural language and deep learning data. Also, these evaluation criteria are connected with several distinguishable GAN-based oversampling methods, generating a cWGAN with a Gumbel softmax distribution that prompts the proper domain to be considered as a DGA, and these results are specific words. Certain DGA words outperform detections and extremely advanced adversarial DGAs by the rate of DGA success attack, word repetition, and domain collision before the creation approaches. Our experimentation with WordDGA validates that wordlist-based DGAs are better than a DGA classifier for identification. Experiments implemented four dependently collected character-based DGAs, word-based DGAs, and mixed DGAs. These outcomes of complicated real-world scenarios showed further techniques of detection rate realization that improved outcomes, AUC-precision, and each of the three datasets and reached a better result regarding these wordlist predictions. Adversarial DGAs and loaded datasets and distinguishable data out of the varied authorities have been operated through the various sub-categories of DGA. Its changes may create complicated experimental results. Our work and the proposed examples correctly identify characters and words, except for some subordinate phrases and mixed DGAs, which is a DNS approach in domain standards. Of the 3–7 g embeddings, the integrated-gram embedding produced the most extraordinary F1-score result. In summary, the optimal way to build word vectors for DGA domain classification using a hybrid layer is to extract sequences of characters and words from domain names. Without manually defining and creating domain characteristics, the proposed model was able to increase the accuracy of DGA domain recognition tasks and obtain better results, not just on the pseudo-random text that DGA types generate. Afterwards, appending for good measure vocabulary, solving the condition of the word phrases may significantly reduce n-distribution. In the future, the authority of these features will be enhanced severely through advanced real-world frameworks, and the training and testing time complexity of DGA botnets will be lowered.

Author Contributions: Conceptualization, S.S. and R.P.; methodology, S.S. and R.P.; software, S.S.; validation, S.S. and R.P.; formal analysis, S.S.; investigation, S.S.; resources, S.S. and R.P.; data curation, S.S.; writing—original draft preparation, S.S. and R.P.; writing—review and editing, S.S. and R.P.; visualization, S.S.; supervision, R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in share the UTL_DGA22 dataset with researchers and scientists in the same field for study, research, supplement, and non-commercial uses. These data were derived from resources available in the public domain and were analyzed in this study, and links are available in the manuscript.

Acknowledgments: We are so thankful to Johannes Bader, Mattia Zago (University of Murcia), and the 360NetLab team for their valuable research into their dataset's DGA botnet and malicious domains. We thank Fraunhofer FKIE and Daniel Plohmann for their generosity in sharing the DGArchive by Fraunhofer FKIE dataset, which includes many new DGA botnet families and Andrey Abakumov (Yandex 360).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a dynamic reputation system for DNS. In Proceedings of the 19th USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010; pp. 273–290.
2. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
3. Yang, L.; Zhai, J.; Liu, W.; Ji, X.; Bai, H.; Liu, G.; Dai, Y. Detecting Word-Based Algorithmically Generated Domains Using Semantic Analysis. *Symmetry* **2019**, *11*, 176. [CrossRef]
4. Usama, M.; Asim, M.; Latif, S.; Qadir, J.; Fuqaha, A.A. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In Proceedings of the 15th International Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 78–83.
5. Plohmann, D.; Yakdan, K.; Klatt, M.; Bader, J.; Gerhards-Padilla, E. A Comprehensive Measurement Study of Domain Generating Malware. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; USENIX Association: Austin, TX, USA, 2016; pp. 263–278.
6. Anderson, H.S.; Woodbridge, J.; Filar, B. DeepDGA: Adversarially tuned domain generation and detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, Vienna, Austria, 28 October 2016; ACM: New York, NY, USA; pp. 13–21.
7. Yun, X.; Huang, J.; Wang, Y.; Zang, T.; Zhou, Y.; Zhang, Y. Khaos: An adversarial neural network DGA with high anti-detection ability. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2225–2240. [CrossRef]
8. Spooren, J.; Preuveneers, D.; Desmet, L.; Janssen, P.; Joosen, W. Detection of algorithmically generated domain names used by botnets: A dual arms race. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; pp. 1916–1923.
9. Sidi, L.; Nadler, A.; Shabtai, A. MaskDGA: An Evasion Attack Against DGA Classifiers and Adversarial Defenses. *IEEE Access* **2020**, *8*, 16158–161592. [CrossRef]
10. Liu, W.; Zhang, Z.; Huang, C.; Fang, Y. CLETer: A character-level evasion technique against deep learning DGA classifiers. *ICST Trans. Secur. Saf.* **2021**, *7*, e5. [CrossRef]
11. Hu, X.; Chen, H.; Li, M.; Cheng, G.; Li, R.; Hua, W.; Yuan, Y. ReplaceDGA: BiLSTM-based Adversarial DGA with high Anti-Detection Ability. *IEEE Tans. Inf. Forensics Secur.* **2023**, *18*, 4406–4421. [CrossRef]
12. Ravi, V.; Alazab, M.; Srinivasan, S.; Arunachalam, A.; Soman, K.P. Adversarial defense: DGA-based botnets and DNS homographs detection through integrated deep learning. *IEEE Trans. Eng. Manag.* **2023**, *70*, 249–266. [CrossRef]
13. Alaeiyan, M.; Parsa, S.; Vinod, P.; Conti, M. Detection of algorithmically-generated domains: An adversarial machine learning approach. *Comput. Commun.* **2020**, *160*, 661–673. [CrossRef]
14. Piras, G.; Pintor, M.; Demetrio, L.; Biggio, B. Explaining machine learning DGA detectors from DNS traffic data. *arXiv* **2022**, arXiv:2208.05285.
15. Hoang, X.D.; Vu, X.H. A Novel Machine Learning-based Approach for Detecting Word-based Botnets. *J. Theor. Appl. Inf. Technol.* **2021**, *99*, 6004–6014.
16. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting Domain Generation Algorithms with Long Short-Term Memory Networks. *arXiv* **2016**, arXiv:1611.00791.
17. Lison, P.; Mavroeidis, V. Automatic Detection of Malware-Generated Domains with Recurrent Neural Models. *arXiv* **2017**, arXiv:1709.07102.
18. Koh, J.J.; Rhodes, B. Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings. In Proceedings of the 2018 IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 2966–2971. Available online: https://ieeexplore.ieee.org/abstract/document/8622066 (accessed on 10 December 2018).
19. Tran, D.; Mac, H.; Tong, V.; Tran, H.A.; Nguyen, L.G. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing* **2018**, *275*, 2401–2413. [CrossRef]
20. Vinayakumara, R.; Somana, K.; Poornachandranb, P.; Kumara, S.S. Evaluating Deep Learning Approaches to Characterize and Classify the DGAs at Scale. *J. Intell. Fuzzy Syst.* **2018**, *34*, 1265–1276. [CrossRef]
21. Xu, C.; Shen, J.; Du, X. Detection method of domain names generated by DGAs based on semantic representation and deep neural network. *Comput. Secur.* **2019**, *85*, 77–88. [CrossRef]

22. Yu, B.; Pan, J.; Hu, J.; Nascimento, A.; De Cock, M. Character Level based Detection of DGA Domain Names. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.
23. Akarsh, S.; Sriram, S.; Poornachandran, P.; Menon, V.K.; Soman, K.P. Deep Learning Framework for Domain Generation Algorithms Prediction Using Long Short-term Memory. In Proceedings of the 5th International Conference on Advanced Computing Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 666–671.
24. Qiao, Y.; Zhang, B.; Zhang, W.; Sangaiah, A.K.; Wu, H. DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism. *Appl. Sci.* **2019**, *9*, 4205. [CrossRef]
25. Liu, Z.; Zhang, Y.; Chen, Y.; Fan, X.; Dong, C. Detection of Algorithmically Generated Domain Names Using the Recurrent Convolutional Neural Network with Spatial Pyramid Pooling. *Entropy* **2020**, *22*, 1058. [CrossRef]
26. Ren, F.; Jiang, Z.; Wang, X.; Liu, J. A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity* **2020**, *3*, 4. [CrossRef]
27. Cucchiarelli, A.; Morbidoni, C.; Spalazzi, L.; Baldi, M. Algorithmically generated malicious domain names detection based on n-grams features. *Expert Syst. Appl.* **2021**, *170*, 114551. [CrossRef]
28. Highnam, K.; Puzio, D.; Luo, S.; Jennings, N.R. Real-Time Detection of Dictionary DGA Network Traffic Using Deep Learning. *SN Comput. Sci.* **2021**, *2*, 110. [CrossRef]
29. Namgung, J.; Son, S.; Moon, Y.S. Efficient Deep Learning Models for DGA Domain Detection. *Secur. Commun. Netw.* **2021**, *2021*, 8887881. [CrossRef]
30. Yilmaz, I.; Siraj, A.; Ulybyshev, D. Improving DGA-Based Malicious Domain Classifiers for Malware Defense with Adversarial Machine Learning. In Proceedings of the IEEE 4th Conference on Information and Communication Technology (CICT), Chennai, India, 3–5 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
31. Yang, C.; Lu, T.; Yan, S.; Zhang, J.; Yu, K. N-trans: Parallel detection algorithm for DGA domain names. *Future Internet* **2022**, *14*, 209. [CrossRef]
32. Vranken, H.P.; Alizadeh, H. Detection of DGA-generated domain names with TF-IDF. *Electronics* **2022**, *11*, 414. [CrossRef]
33. Liew, S.R.C.; Law, N.F. Use of subword tokenization for domain generation algorithm classification. *Cybersecurity* **2023**, *6*, 49. [CrossRef]
34. Keras GitHub Repository. Available online: https://github.com/keras-team/keras (accessed on 1 June 2023).
35. Charan, P.S.; Shukla, S.K.; Anand, P.M. *Detecting Word Based DGA Domains Using Ensemble Models*; Springer International Publishing: Cham, Switzerland, 2020.
36. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved training of Wasserstein GANs. *arXiv* **2017**, arXiv:1704.00028.
37. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
38. Zhang, Y.; Gan, Z.; Fan, K.; Chen, Z.; Henao, R.; Shen, D.; Carin, L. Adversarial feature matching for text generation. In Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017; pp. 4006–4015.
39. Zhang, W.E.; Sheng, Q.Z.; Alhazmi, A.; Li, C.L. Generating Textual Adversarial Examples for Deep Learning Models: A Survey. *arXiv* **2019**, arXiv:1901.06796.
40. Wordninja GitHub Repository. Available online: https://github.com/keredson/wordninja (accessed on 1 June 2023).
41. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the 1st International Conference on Learning Representations (ICLR), Scottsdale, Arizona, 2–4 May 2013.
42. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep Contextualized Word Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Association for Computational Linguistics: New Orleans, LA, USA, 2018; Volume 1, pp. 2227–2237.
43. Zhao, H.; Chang, Z.; Bao, G.; Zeng, X. Malicious domain names detection algorithm based on N-Gram. *J. Comput. Netw. Commun.* **2019**, *2019*, 4612474. [CrossRef]
44. Tranco Website: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. Available online: https://trancolist.eu/ (accessed on 1 June 2023).
45. Bader, J. Domain_Generation_Algorithms Repository. GitHub, 2018. Available online: https://github.com/baderj/domain_generation_algorithms (accessed on 20 August 2023).
46. DGArchive Repository Access Portal. Available online: https://dgarchive.caad.fkie.fraunhofer.de/welcome/ (accessed on 1 June 2023).
47. Zago, M.; Pérez, M.G.; Pérez, G.M. UMUDGA: A dataset for profiling DGA-based botnet. *Comput. Secur.* **2020**, *92*, 101719. [CrossRef]
48. OSINT. Feeds from Bambenek Consulting. 2021. Available online: https://osint.bambenekconsulting.com/feeds/ (accessed on 15 March 2023).
49. Network Security Research Lab at 360. Netlab DGA Project. Available online: https://data.netlab.360.com/dga/ (accessed on 11 March 2023).
50. Tuan, T.A.; Anh, N.V.; Luong, T.T.; Long, H.V. Utl_dga22-a dataset for dga botnet detection and classification. *Comput. Netw.* **2023**, *221*, 109508. [CrossRef]

51. Jenks, G. Python Word Segmentation. Available online: https://github.com/grantjenks/python-wordsegment (accessed on 10 June 2022).
52. Hwang, C.; Kim, H.; Lee, H.; Lee, T. Effective DGA-Domain Detection and Classification with TextCNN and Additional Features. *Electronics* **2020**, *9*, 1070. [CrossRef]
53. Adam Optimizer (Keras Documentation). Available online: https://keras.io/api/optimizers/adam/ (accessed on 1 June 2023).
54. Vij, P.; Nikam, S.; Bhatia, A. Detection of Algorithmically Generated Domain Names using LSTM. In Proceedings of the International Conference on COMmunication Systems and NETworkS (COMSNETS), Bengaluru, India, 7–11 January 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
55. Yadav, S.; Reddy, A.K.K.; Reddy, A.L.N.; Ranjan, S. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *IEEE ACM Trans. Netw.* **2012**, *20*, 1663–1677. [CrossRef]
56. Antoniou, A.; Storkey, A.; Edwards, H. Data augmentation generative adversarial networks. *arXiv* **2017**, arXiv:1711.04340.
57. Wang, T.; Chen, L.; Genc, Y. A dictionary-based method for detecting machine-generated domains. *Inf. Secur. J. Glob. Perspect.* **2020**, *30*, 205–218. [CrossRef]
58. Patsakis, C.; Casino, F. Exploiting statistical and structural features for the detection of domain generation algorithms. *J. Inf. Secur. Appl.* **2021**, *58*, 102725. [CrossRef]
59. Yang, L.; Liu, G.; Zhai, J.; Dai, Y.; Yan, Z.; Zou, Y.; Huang, W. A novel detection method for word-based DGA. In Proceedings of the International Conference on Cloud Computing and Security, Haikou, China, 8–10 June 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 472–483.
60. Zebin, T.; Rezvy, S.; Luo, Y. An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2339–2349. [CrossRef]
61. Sheatsley, R.; Papernot, N.; Weisman, M.J.; Verma, G.; McDaniel, P. Adversarial examples for network intrusion detection systems. *J. Comput. Secur.* **2022**, *30*, 727–752. [CrossRef]