Check for updates

Hindawi
Security and Communication Networks
Volume 2021, Article ID 8887881, 15 pages
https://doi.org/10.1155/2021/8887881

WILEY | Hindawi

Research Article

Efficient Deep Learning Models for DGA Domain Detection

Juhong Namgung, Siwoon Son, and Yang-Sae Moon

Dept. of Computer Science, Kangwon National University, Chuncheon-si, Gangwon-do 24341, Republic of Korea

Correspondence should be addressed to Siwoon Son; ssw5176@kangwon.ac.kr

Received 6 August 2020; Revised 17 December 2020; Accepted 7 January 2021; Published 19 January 2021

Academic Editor: Savio Sciancalepore

Copyright © 2021 Juhong Namgung et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, cyberattacks using command and control (C&C) servers have significantly increased. To hide their C&C servers, attackers often use a *domain generation algorithm* (*DGA*), which automatically generates domain names for the C&C servers. Accordingly, extensive research on DGA domain detection has been conducted. However, existing methods cannot accurately detect continuously generated DGA domains and can easily be evaded by an attacker. Recently, long short-term memory- (LSTM-) based deep learning models have been introduced to detect DGA domains in real time using only domain names without feature extraction or additional information. In this paper, we propose an efficient DGA domain detection method based on bidirectional LSTM (BiLSTM), which learns *bidirectional* information as opposed to *unidirectional* information learned by LSTM. We further maximize the detection performance with a convolutional neural network (CNN) + BiLSTM ensemble model using Attention mechanism, which allows the model to learn both local and global information in a domain sequence. Experimental results show that existing CNN and LSTM models achieved F1-scores of 0.9384 and 0.9597, respectively, while the proposed BiLSTM and ensemble models achieved higher F1-scores of 0.9618 and 0.9666, respectively. In addition, the ensemble model achieved the best performance for most DGA domain classes, enabling more accurate DGA domain detection than existing models.

1. Introduction

Despite advances in security technology, it is still difficult to efficiently respond to cyberattacks using command and control (C&C) servers. A C&C server generally controls botnets, a set of infected PCs, issuing malicious commands, or controlling malicious code from remote attackers. Representative cyberattacks using C&C servers include spreading malware to infected PCs and DDoS attacks through botnets. Because the malware on an infected PC is installed at a remote location, it attempts to connect to the C&C server to receive commands. Thus, the malware must know the IP address or domain name of the C&C server. The simplest way to achieve this is to fix a hard-coded IP address or domain name of the C&C server for the malware. However, this is not effective from an attacker's perspective because it is easily detected and blocked by security experts.

Attackers often use a *domain generation algorithm* (*DGA*) to defend and evade the domain of their C&C server from being blocked. A DGA continuously generates multiple domain names using a certain seed [1]. The malware assumes that the domain generated by the DGA is the domain of the C&C server and attempts to connect to the C&C server. Thus, it is very important to detect the dynamic domain of the C&C server because the DGA constantly changes the domain name. It is also important to classify which DGA generates the domain because DGAs have different purposes and types, such as ransomware and bank trojans. In addition, because malicious domains (i.e., DGA domains) and normal domains (i.e., non-DGA domains) have a multiclass imbalance problem with very different incidences for each class, their classification is a particularly challenging problem.

The simplest way to detect DGA domains is to use a blacklist. However, this is neither exhaustive nor effective because a large number of new domains are constantly generated [2]. Reverse engineering is also often used to deconstruct a DGA, which identifies the DGA from

malicious domain samples and then blocks that DGA's domains. However, reverse engineering only prevents a small part of a DGA and requires significant time and effort from domain experts. Thus, the reverse engineering method is very slow and impractical compared to other methods.

As a countermeasure, machine learning-based methods that consider lexical features have appeared. Representatively, there has been an attempt to exploit statistical features, such as length, bigram, and entropy of domain characters [3], as well as an attempt to apply a lexical pattern extracted from non-DGA domains to machine learning algorithms, such as random forests [4]. However, the feature extraction process also requires significant time and expertise, and attackers can identify features to evade these detection methods. Other recent studies have used additional information to improve performance. For example, WHOIS information is frequently used as side information, including domain owner contact, domain availability status, and domain registration company. However, these methods cannot detect DGA domains in real time. In addition, since the General Data Protection Regulation (GDPR) came into force, it is no longer possible to obtain WHOIS information [5].

To solve the problems of existing DGA domain detection methods, many deep learning-based methods that improve performance without additional information have recently been proposed [6–12]. These deep learning-based methods do not require a time-consuming feature extraction process, and as black boxes, it is difficult for attackers to perform reverse engineering or evade these methods. These methods can also detect DGA domains in real time using only the domains without additional information. Among various deep learning models, recurrent neural networks (RNNs) with high accuracy for sequence data have been widely used. Because domain names can be expressed as sequences of characters, RNN models have higher performance than other neural network models. In particular, an RNN-based long short-term memory (LSTM) is suitable for solving the class imbalance problem of DGA domain detection [6].

In this study, we aimed to improve the performance of deep learning models for DGA domain detection. First, we propose a model that uses bidirectional LSTM (BiLSTM), which can learn *bidirectional* information by extending the existing *unidirectional* LSTM model. The existing LSTM model learns only forward information from the past to the future based on the current time point. BiLSTM [13] complements this by also learning backward information from the future to the past. In this paper, we present the detailed design and implementation of a new DGA domain detection method using the proposed BiLSTM model. Then, we propose a further improved DGA domain detection method based on an ensemble model that combines a convolutional neural network (CNN) and BiLSTM with Attention. Existing deep learning-based methods generally use only one model of CNN or LSTM, each of which has its own characteristics and discriminative performance. By using an ensemble technique [14] that trains multiple models and uses the results of those models together, we can gain the advantages of eac h model and significantly improve the performance.

Our contributions can be summarized as follows:

(1) We analyze case studies of existing deep learning-based DGA domain detection methods in detail

(2) We extend DGA domain detection from *binary* classification to *multiclass* classification to determine the type of DGA domain, not just whether it is a DGA domain

(3) To classify DGA domains, we propose both a novel method of using a BiLSTM model that extends the existing LSTM model and an optimized CNN + BiLSTM ensemble method that combines CNN and BiLSTM models

(4) We design and implement DGA domain detection methods using existing and proposed models and demonstrate the superiority of the proposed methods through extensive performance comparison using the latest and most realistic datasets

The remainder of this paper is organized as follows. In Section 2, the DGA and related work on DGA domain detection are analyzed. In Section 3, the theoretical background of deep learning models is explained. In Section 4, the two proposed efficient detection methods in detail with existing models are presented. In Section 5, the overall procedure of DGA domain detection is described, and in Section 6, the detection performance of the proposed deep learning models is experimentally evaluated. Finally, in Section 7, the conclusions of the study are presented.

2. Related Work

There have been several studies to detect the DGA domain [6–12, 15–18]. In this section, we have described and compared the existing DGA domain detection methods in detail for continuity of the study. Section 2.1 describes the working procedure of the DGA and representative DGA examples. Section 2.2 describes the existing feature-based DGA domain detection methods and their limitations. In particular, it provides full details of how to use statistical features and how to use additional information. Section 2.3 describes deep learning-based DGA domain detection methods using CNN and RNN models.

2.1. Domain Generation Algorithm. An attacker uses a DGA to prevent their C&C server, which is controlling botnets, from being blocked. Figure 1 illustrates the working of a DGA. In Step 1, the attacker generates domain names using the DGA and a seed and registers one of them as the domain name of the C&C server. In Step 2, the malware also generates a domain name with the same DGA and seed. In Step 3, the malware queries the DNS server for the generated domain name and requests the IP address of the C&C server. If the domain name is not registered, the DNS server responds with NXDomain (Non-eXistent Domain), but the malware continues to query the newly generated domain name from the DNS server until it is able to connect to the C&C server. Eventually, in Step 4, the malware that obtains a

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

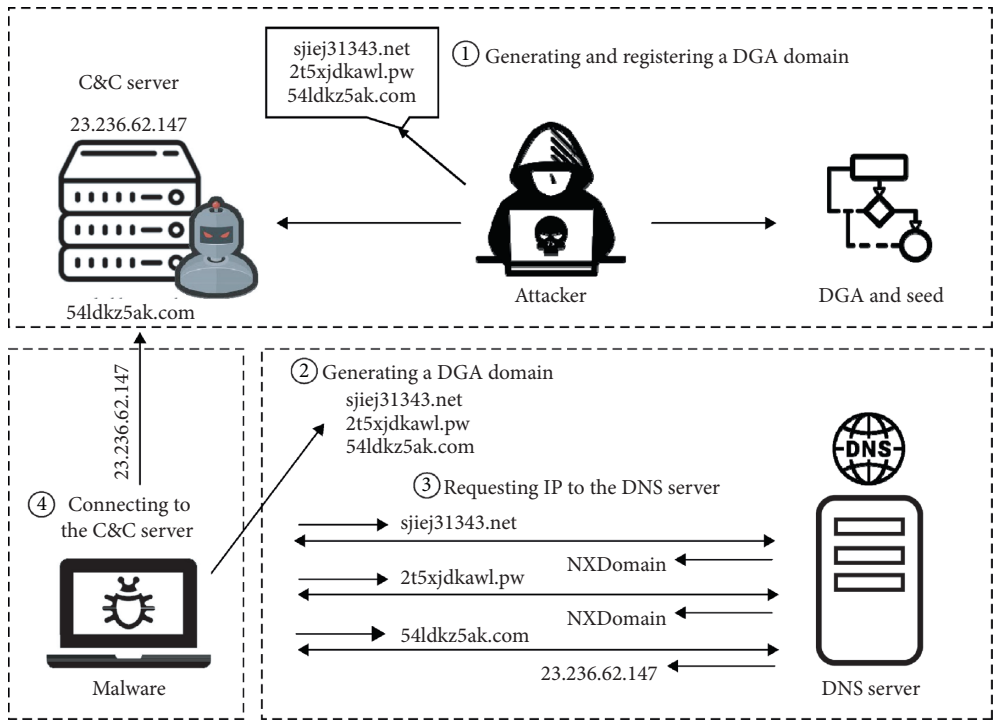

Figure 1: Attacker's procedure for using a DGA.

valid IP address of the C&C server tries to communicate with the C&C server. The DGA changes the domain name occasionally so that the attacker can easily hide the C&C server. Recently, DGAs have become more intelligent. For example, Torpig uses Twitter's latest topics as DGA seeds, and Kraken uses sophisticated word generators to imitate a normal domain name [19]. Therefore, it is very challenging to detect such developing DGA domains.

In addition, various DGA types exist depending on their purpose. Table 1 lists various DGA types and corresponding examples of generated domains. As the most representative DGA, Banjori is used for stealing information from online banking users [20]. According to NSFOCUS, Banjori was first discovered in 2013; for one year in 2019, a total of 1,499 related botnets had been detected, and this number is still increasing [20]. In addition, DGAs designed to steal financial institution information or infect with worms, such as Tinba and Ramnit, exist [21]. Because various DGA types exist, it is also important to classify the exact types of DGA domains.

2.2. Feature-Based DGA Domain Detection. To overcome the shortcomings of early blacklist and reverse engineering methods, many studies have attempted to use statistical features of domain names. Yadav et al. [15] used the bigram and distribution of characters and numbers within domain names to distinguish between DGA and non-DGA domain clusters. They compared three mathematical statistics often used to cluster domains: Kullback–Leibler, Jaccard index, and Edit distance. Wei-wei et al. [16] detected DGA domains by considering the morpheme and lexical features of domain names. In addition to existing lexical features, they proposed a heuristic approach to find morpheme tokens in domain names and consider the features of these tokens. However, these manual feature extraction methods are both time-consuming and can be easily avoided by attackers.

Other studies have used supplementary information in addition to the statistical features of domains. Curtin et al. [7] used WHOIS data as additional information, including availability, creation date, and contact information of the domain names. Schiavoni et al. [17] detected DGA botnets using IP-based features, a blacklist, and DNS query information as well as linguistic-based features. Antonakakis et al. [18] detected DGA botnets by analyzing NXDomain responses that occurred when accessing nonexisting domains. In general, bots from the same botnet generate similar NXDomain traffic. Based on this concept, the researchers clustered the domains by considering the similarity between domain names and DNS responses. They then determined whether they were DGA-generated or not through alternating decision trees. However, the EU recently drafted and passed the GDPR, which regulates how data are processed to protect data privacy. As a result, DNS Belgium, which was responsible for managing the top-level domain in Belgium, changed the WHOIS system so that personal information, such as e-mail addresses, is no longer displayed on the Internet. Therefore, these methods using additional information can no longer be used, even to provide detection services, because of the GDPR.

2.3. Deep Learning-Based DGA Domain Detection. In recent years, several studies on deep learning have considered DGA domain detection. Feng et al. [8] used five representative

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 1: Representative DGA types and example domains.

Type	Purpose	Examples
Non-DGA	Normal domain	(i) google.com (ii) youtube.com (iii) tmall.com
Banjori	Banking trojan	(i) ffpienhancedysb.com (ii) mytfererwyatanb.com (iii) oefdmen.com
Tinba	Malware targeted to financial institutions	(i) mgosynrcfulu.ru (ii) sjmhsruveydg.pw (iii) wgnmqphhgtim.net
Ramnit	Worm targeted to Windows	(i) evorcdtbnrnosafjqeb.eu (ii) fuxwslcu.com
Necurs	Ransomware distributor	(i) adydlcbumuoncy.com (ii) lyxvfcgtdtfavescbvoo.eu

CNN-based image classification models for DGA domain classification: Alex Net, VGG, Squeeze Net, Inception, and ResNet. Their experimental results showed high performance even though they used image-specific models, particularly the Inception v4 model. Yu et al. [9] compared stacked CNN and parallel CNN models using DGA domain characters as input. Their experimental results showed that both CNN models outperformed machine learning models, such as random forest and multilayer perceptron, particularly the parallel CNNs, which had higher accuracy.

RNN models, and in particular LSTM models, have also been studied in the field of DGA domain detection. Woodbridge et al. [10] classified DGA detection into two categories—retrospective detection and real-time detection—and they proposed an LSTM model for real-time detection. Experimental results showed that the performance of the LSTM model without feature extraction was better than that of a random forest with manual feature extraction. Qiao et al. [11] applied an Attention mechanism to an existing LSTM model and showed that the importance of each character in the domain name was different in DGA domain classification. They experimentally classified 15 types of DGA and non-DGA domain names and showed that their method outperforms not only legacy machine learning models but also LSTM models. Although there have been many studies on detecting DGA domains by applying LSTM [6, 7, 10, 12], they all use solely LSTM with unidirectional information, which is a major limitation because they do not use all available input information in the current state. In this paper, we propose a BiLSTM-based detection model that exploits high-performance LSTM models but learns bidirectional information. We also propose a CNN + BiLSTM ensemble model for better DGA domain detection performance.

The main difference between the existing deep learning-based methods and the proposed methods lies in the underlying deep learning models. As deep learning models have different purposes, their working procedure is also different. Table 2 presents a comparison of the deep learning models used for DGA domain detection in terms of target features. First, a CNN, which is used mainly for image processing, can extract features from local spatial information, but does not consider any temporal information. Next, the LSTM suitable for time series data has the ability to process temporal information, but does not consider local spatial information. To compensate for this, there is a recent case of introducing an Attention mechanism to LSTM to focus on specific characters in the domain. However, as LSTM learns only in the forward direction, it can still partially process temporal information. The BiLSTM introduced in this paper completely processes temporal information by learning in the backward direction to overcome the limitations of the existing LSTM. Furthermore, the ensemble model of CNN and BiLSTM proposed to consider local spatial information can yield the highest performance gain by extracting most of the features from the same data.

3. Theoretical Background

This section describes the theoretical concepts of deep learning models mainly used in DGA domain detection. Existing deep learning-based DGA domain detection methods generally use only domain names, and they can be classified as CNN or RNN schemes. Because a domain can be considered as a sequence of characters, several studies have exploited RNNs, particularly with LSTM models. Furthermore, there have been a few recent attempts to add an Attention mechanism to LSTMs. Thus, in this section, we describe the concepts of CNNs, LSTM, and the Attention mechanism.

3.1. CNN. A CNN is a deep learning model that shows excellent performance in the field of image recognition by using local region features. A CNN consists of a feature extraction neural network and a classification neural network. First, the feature extraction step repeatedly performs convolutional and pooling layers, automatically extracting the data features. As the core part of a CNN, the convolutional layer maps multiple filters (or kernels) to the input data to find the appropriate filter (i.e., to find features well). It creates feature maps by moving the filters by a certain window size and performing a convolutional operation between them and the entire input data. The pooling

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 2: Comparison of deep learning models used in existing and proposed approaches.

Target features	Existing approaches			Our approaches	
	CNN [8, 9]	LSTM [6, 7, 10, 12]	LSTM with attention [11]	BiLSTM with attention	Ensemble (CNN + BiLSTM)
Local spatial information	x̂	×	×	×	x̂
Global spatial information	×	x̂	x̂	x̂	x̂
Weighted spatial information	×	×	x̂	x̂	x̂
Forward temporal information	×	x̂	x̂	x̂	x̂
Backward temporal information	×	×	×	x̂	x̂

layer reduces the size of the feature map or obtains a representative value from the feature map. The representative methods used in this pooling layer are max pooling and average pooling.

Next, the CNN outputs the extracted features to the classification step, a fully-connected layer, which transforms the two-dimensional features extracted from the convolutional layer into a 1-dimensional vector. With the transformed input vector, each neuron in the fully connected layer is connected to all neurons in the previous layer. Finally, the last layer outputs the probabilities of the classification classes.

3.2. LSTM. LSTM is the most used RNN-based model for DGA domain detection. In LSTM, an RNN uses the output of the previous step in the training of the current step, which is very suited to the processing of sequential data such as speech or text. An RNN can be expressed by equations (1) and (2):

$$h_t = \sigma\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right), \tag{1}$$

$$y_t = W_{hy}h_t + b_y. \tag{2}$$

In these equations, h_t denotes a hidden state at time t, where x and y are the input and output, respectively. In equation (1), h_t is calculated as the input x_t of the current state and the previous hidden state h_{t-1}. σ is an activation function, and W_{xh}, W_{hh}, and b_h are parameters to be learned. In equation (2), the predicted value y_t is calculated using h_t, and W_{hy} and b_y are learning parameters.

However, this RNN model may cause a vanishing/exploding gradient problem due to the gradient becoming too small or too large during backpropagation in a deep layer. LSTM has appeared to solve this long-term dependency problem by adding a cell state that effectively conveys previous learning results. Figure 2 shows the internal architecture of LSTM. LSTM is an RNN-based chain structure that conveys previous information through a cell and determines the cell state through three gates: *forget*, *input*, and *output*.

LSTM is expressed by equations (3) to (7):

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \tag{3}$$

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right), \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \tag{5}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right), \tag{6}$$

$$h_t = o_t \tanh\left(c_t\right). \tag{7}$$

First, the forget gate f determines which information to discard (forget) from the cell state. Next, the input gate i, which stores the current information, determines which information to update in the cell state among the new inputs. Then, it creates a new candidate for delivery to the cell through the tanh layer. Subsequently, the information generated through the forget and input gates is combined to update the cell state. Finally, the output gate o determines which part of the cell state to output, and the final result value h_t is determined through the cell state output and the output gate.

3.3. Attention Mechanism. Recently, in machine translation, the Attention mechanism [22] has been widely used with LSTM. Machine translation often uses the sequence-to-sequence (seq2seq) model to transform an input sequence into a target sequence. The seq2seq model generally has an encoder-decoder structure and consists of two RNNs, e.g., in LSTM. The encoder compresses the input sequence into a fixed-length context vector. Because this method expresses the entire input sequence as a single vector, information loss may occur when the input sequence is long. The Attention mechanism, which emerged to solve the information loss problem, allows the model to focus on the important parts. In particular, it transfers information related to the more relevant words (or characters in the domain name) in the prediction directly to the decoder.

Figure 3 illustrates the architecture of the Attention mechanism, which can be mathematically expressed by equations (8)–(11):

$$s_i = f\left(s_{i-1}, y_{i-1}, c_i\right), \tag{8}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij}h_j, \tag{9}$$

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

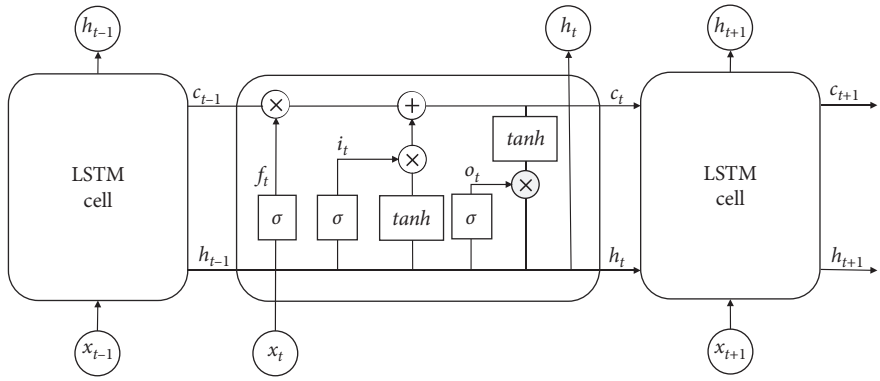

Figure 2: Internal architecture of an LSTM model.

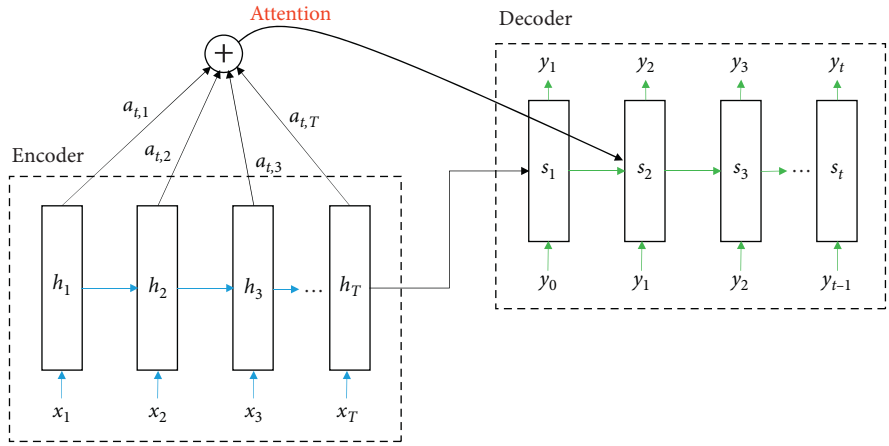

Figure 3: General architecture of the attention mechanism.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{10}$$

$$e_{ij} = a(s_{i-1}, h_j) = v_a^T \tanh(W_a[s_{i-1}, h_j]). \tag{11}$$

First, the encoder receives the input sequence x_1, x_2, ..., x_T and generates the corresponding hidden state vector h_i. Then, as shown in equation (8), the current hidden state s_i of the decoder is calculated as the previous hidden state vector s_{i-1}, the previous decoder output value y_{i-1}, and the current context vector c_i. At this time, the context vector is calculated through the weighted sum of the Attention probability values a_{ij} of the encoder, as shown in equation (9). Here, T_x is the length of the encoder input word. This means that the decoder considers all inputs of the encoder, as shown in Figure 3. In equation (10), the attention probability is calculated as a normalized softmax function of the attention score e_{ij}, which is a scalar value indicating how similar the previous hidden state vector s_{i-1} is to the encoder's j-th vector h_j. Equation (11) calculates the actual similarity using tanh, where v and W are the learning parameters for applying the attention.

4. Deep Learning Models for DGA Domain Detection

This section describes deep learning models for DGA domain detection in detail. We first present the existing CNN and LSTM models in Sections 4.1 and 4.2, respectively. We then propose a BiLSTM-based model in Section 4.3 and a CNN + BiLSTM ensemble model in Section 4.4.

4.1. CNN-Based DGA Detection Model. A CNN is generally used for image data processing, but it can also process one-dimensional (1D) sequence data; such a CNN is called a 1DCNN, which can solve a classification problem by extracting features from sequential data. 1DCNNs have the advantage of being able to learn local information; thus, they are mainly used for sensor data or natural language processing. For our application, we consider the domain name as a sequence of characters to use a 1DCNN for DGA domain classification.

We use a parallel CNN model [23] to apply different kernel sizes of the 1DCNN and sum the outputs. The model improves performance by extracting features from various kernel sizes. In this study, we choose the optimal structure

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

and model parameters by referring to the existing parallel CNN model-based method [9]. Figure 4 illustrates the architecture of the CNN model. First, in the feature extraction step, we apply four 1D-convolutional layers with different kernel sizes from 2 to 5, where the number of filters is always 256. We apply a max pooling layer after the 1D-convolutional layer, similar to the existing CNN models. Subsequently, in the classification step, we concatenate the four 1DCNN results and stack three hidden layers to finally output the classification result. At this time, each hidden layer comprises a fully connected layer, activation layer, batch normalization, and dropout. The following gives a detailed description of the hidden layer. First, we use an exponential linear unit, which complements the problems with using a rectified linear unit (ReLU) as the activation function. Next, batch normalization enables the results of the activation function to follow a normal distribution; thus, the model can learn without being heavily dependent on the initial weight. Finally, the dropout omits neurons with a certain probability rather than learning all neurons to prevent overfitting. We set the dropout rate to 0.5.

4.2. LSTM-Based DGA Detection Model. Various DGA domain detection methods have shown notable performance improvements through the use of an LSTM model [6, 7, 10, 12], and recently, the Attention mechanism has also been used together with LSTM models. Qiao et al. [11] found a case where only the first four characters in the domain names generated by a specific DGA were changed, while the remainder were unchanged. This meant that they only had to focus on a specific part of the entire domain name, and detection performance could be improved by applying LSTM with the Attention mechanism (which we refer to as LSTM with Attention) to DGA domain detection.

In this study, we built an existing model [11] using LSTM with Attention for comparison, as shown in Figure 5. This LSTM with Attention model comprises an LSTM layer, an attention layer, and a fully connected layer. The model parameters used here are the same as the optimal parameters used in the existing LSTM-based methods [10–12]. First, we use the LSTM layer with an output dimension and configure the return_sequence parameter to true to use the entire sequence information. Next, we use self-attention [22], which recently showed the best performance among various Attention mechanisms. We configure the output size of the attention layer to 128, the same as that of the LSTM layer. Finally, the results of the attention layer are flattened and conveyed to the fully connected layer. As parameters of the fully connected layer, we configure the activation function to ReLU and the dropout rate to 0.5.

4.3. BiLSTM-Based DGA Detection Model. In this section, we propose a BiLSTM with Attention-based DGA detection method that extends the existing LSTM with Attention model. A bidirectional RNN (Bi-RNN) [13] learns through two separate forward and backward RNNs. The Bi-RNN improves the RNN performance because it can check not only forward information but also backward information. A Bi-RNN can be expressed by equations (12) to (14):

$$\overrightarrow{h}_t = \sigma\left(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right), \tag{12}$$

$$\overleftarrow{h}_t = \sigma\left(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right), \tag{13}$$

$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y. \tag{14}$$

In these equations, $\overrightarrow{h}_t$ denotes an RNN connected in the forward direction (i.e., left to right), and h_t denotes an RNN connected in the backward direction (i.e., right to left). That is, a Bi-RNN concatenates the hidden states in both directions to determine the output value y_t.

BiLSTM uses LSTM analogously to the RNNs in a Bi-RNN. The BiLSTM model solves the problem of RNNs with LSTM while simultaneously improving the performance by reflecting bidirectional information. Thus, we propose a BiLSTM model that extends unidirectional LSTM for DGA domain detection. We also apply the Attention mechanism to the BiLSTM model to further improve its performance. The proposed model can solve the problem with the existing LSTM, which is that past data are not considered effectively; further, it can focus on important characters in domain names using the Attention mechanism.

Figure 6 shows the conceptual architecture of the proposed BiLSTM with Attention model. Compared to the LSTM with Attention model in Figure 5, we note that BiLSTM uses forward and backward LSTM models together. The proposed model consists of a BiLSTM layer, attention layer, and fully-connected layer. We set each LSTM output of the BiLSTM layer to 128 dimensions and configure the return_sequence to true to output for each sequence. We use self-attention in the same way as in the LSTM with Attention model after the BiLSTM layer. At this time because the BiLSTM uses two LSTMs, the output size of the BiLSTM layer is 256, which is twice the output size of 128 of the existing LSTM model. Considering that the output sizes of the attention layer and LSTM are equal to 128 in the LSTM with Attention model, we configured the output sizes of the attention layer and BiLSTM layer to 256 in our model. Finally, the results of the attention layer are flattened and input to the fully connected layer. The activation function used in the fully connected layer is a ReLU and the dropout ratio is 0.5, which is the same as in the existing LSTM with Attention model.

4.4. CNN + BiLSTM Ensemble-Based DGA Detection Model. The ensemble approach combines multiple models to improve performance. Representative ensemble methods include bagging, boosting, and stacking [24]. The bagging method trains several models through data samples and then obtains the final results through aggregation, such as averaging and voting. Unlike the bagging method, which trains the model independently, the boosting method focuses on

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

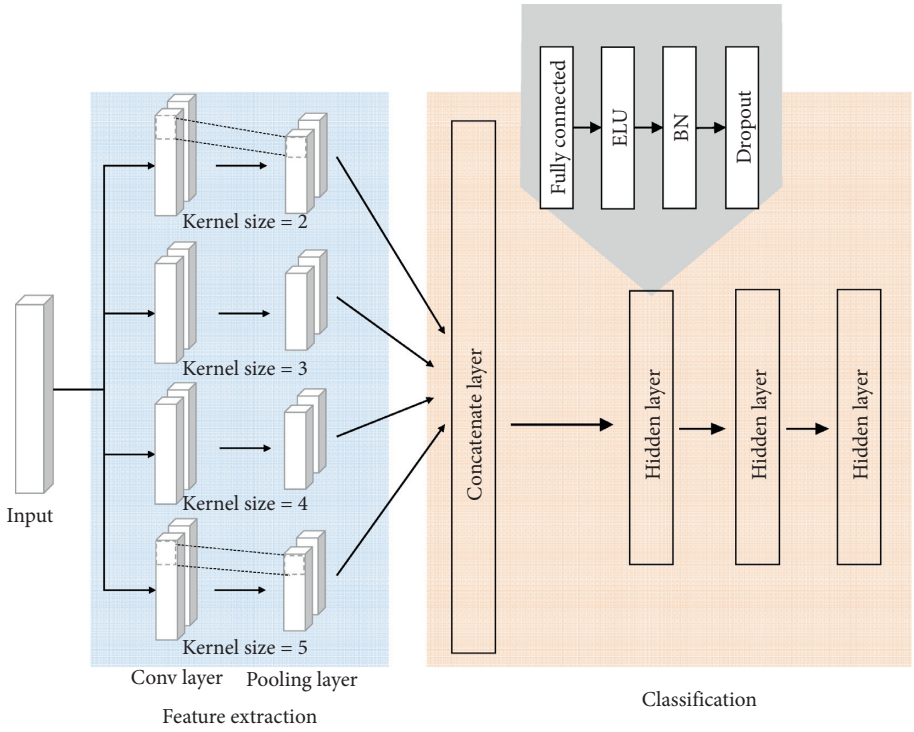

Figure 4: Conceptual architecture of the CNN model.

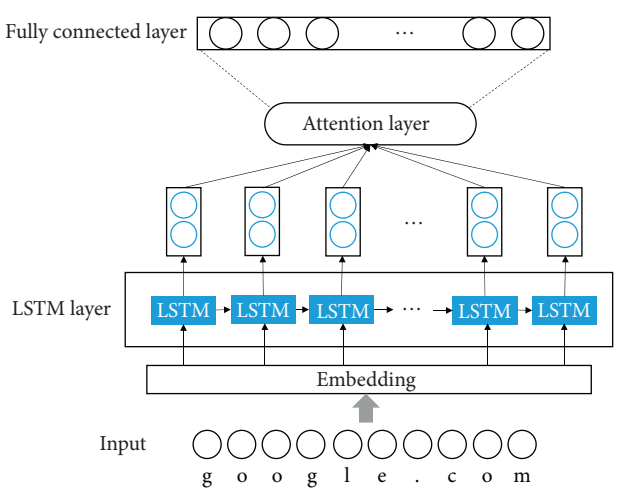

Figure 5: Conceptual architecture of the LSTM with Attention model.

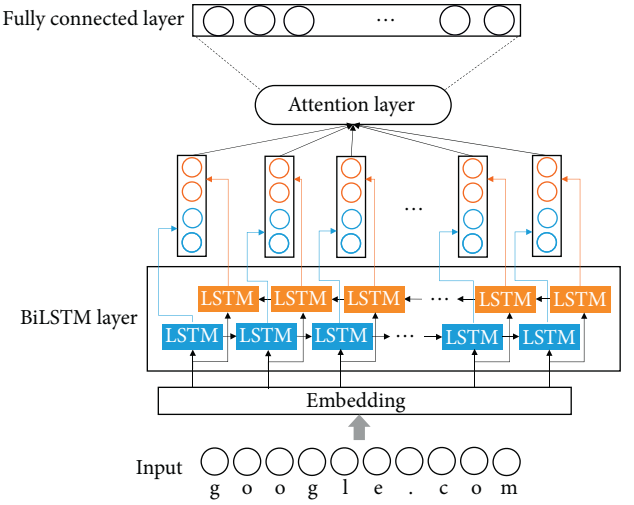

Figure 6: Conceptual architecture of the BiLSTM with Attention model.

misclassified data and trains the models according to weight. The boosting method trains models sequentially, focusing more on the next model by assigning high weights to the misclassified data. The stacking method learns by passing the prediction results of one model to another model. The stacking method can use a variety of algorithms together, so it can take advantage of several models. These ensemble methods are characterized using all the final outputs of the candidate models.

In this paper, we propose a hybrid approach using bagging and stacking methods. The proposed method ensembles the intermediate values of the candidate models through a fully connected layer. That is, the fully connected layer is used for both aggregation of the bagging method and a successive model of the stacking method. We use the CNN and BiLSTM with Attention models as candidate ensemble models. This is an intuitive strategy to take advantage of both a CNN, which learns local sequence information, and BiLSTM, which learns global sequence information. Figure 7 illustrates the conceptual architecture of the proposed ensemble model. First, we perform preprocessing and embedding of the input domain name by applying the CNN layer and BiLSTM with Attention layer, respectively. At this time, we apply the four 1D-convolutional operations with

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

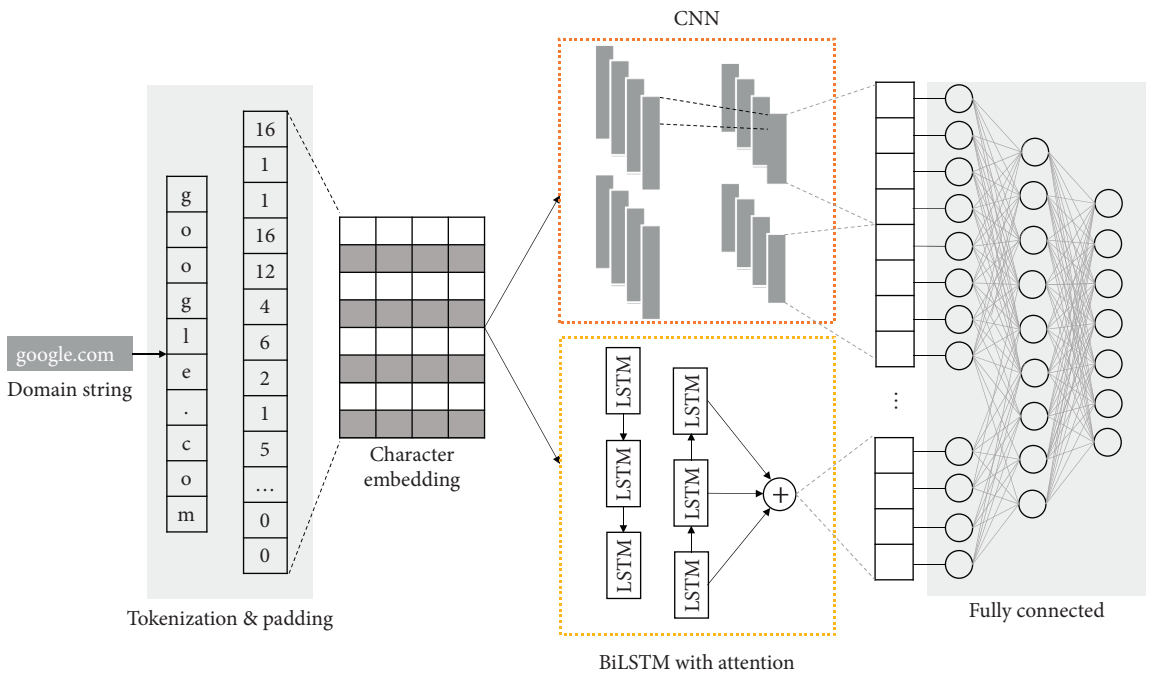

Figure 7: Conceptual architecture of the CNN + BiLSTM ensemble model.

different filter sizes from 2 to 5 and max pooling to the CNN layer, as described in Section 4.1. In addition, the BiLSTM with Attention layer consists of the BiLSTM layer and attention layer, to which self-attention is applied, as described in Section 4.3. After concatenating the outputs of the CNN and BiLSTM with Attention layers, the final DGA class is determined through the fully connected layer. The proposed ensemble model trains in an end-to-end manner by using the intermediate value of each candidate model as the input of the fully connected layer rather than by learning the candidate models separately, similar to in legacy ensemble models. In this manner, the ensemble method can properly reflect the outputs of the CNN and BiLSTM with Attention layers.

5. Overall Procedure of Deep Learning-Based DGA Domain Detection

Figure 8 illustrates the overall process of detecting DGA domains using deep learning models. First, for the given domain names consisting of characters, we perform preprocessing and embedding so that the deep learning models can accept them. The preprocessing consists of tokenization and padding. Next, after the embedding of each domain, we deliver them to the deep learning models presented in Section 4. Finally, the fully connected layer and softmax function classify DGA classes, which include a non-DGA class.

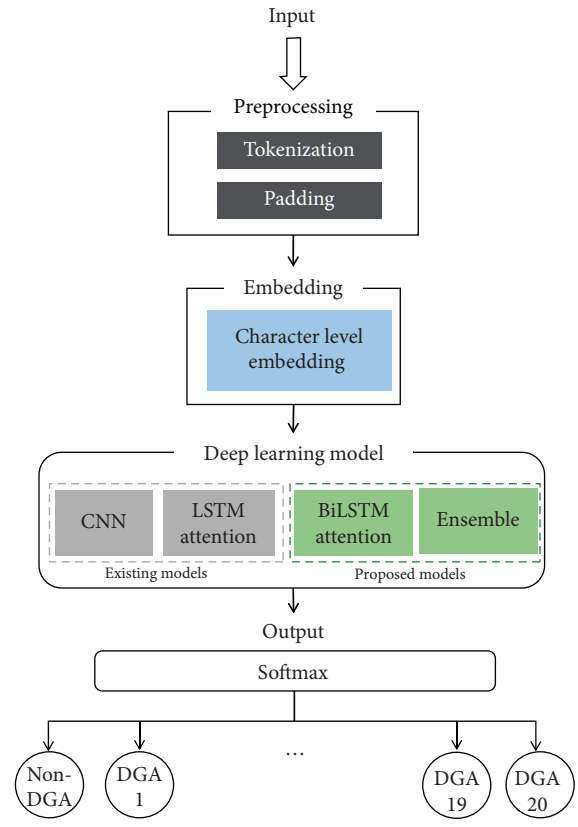

Figure 8: Overall procedure of deep learning-based DGA domain detection.

5.1. Preprocessing. We have two preprocessing steps: (1) a tokenization that replaces the characters in domain names with numbers and (2) a padding that equalizes the input lengths of the domain names. First, the tokenization replaces each character of a domain name with a number for input to the deep learning model. In this study, we used the Tokenizer of Keras [25] to map alphabetical characters (a to z),

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

numbers (0 to 9), and special characters (., −, and _) to integers 0 to 39. Here, the alphabetical characters are not case-sensitive according to the domain generation conditions.

Second, padding equalizes the different lengths of domain names. Figure 9 shows the length distribution of the DGA and non-DGA domain names. As shown in the figure, the lengths of domain names are widely distributed from 4 to 73. In this study, we applied zero padding to a length of 73, the maximum length of any DGA domain, to consider as much information as possible from the domain names.

5.2. Embedding. Embedding maps the input integers (i.e., preprocessed domain names) to a dense vector, which is updated in the deep learning models. Because a domain name is a sequence of characters, we use a character-level embedding method to embed each character in the domain names. Recently, in natural language processing, word-level embedding has shown good performance. However, domain names are generally short and are not made up of words. Thus, for domain names, it is preferable to use character-level embedding rather, contrary to general natural language processing.

The parameters for applying character-level embedding used in this study were as follows. We set the input size of embedding to 73, the same as the padding size, and the output dimension to 32. We also applied dropout and L2 regularization to prevent overfitting. Therefore, as a result of embedding, each domain name was represented by 73×32 dimensional vectors.

5.3. Deep Learning Model. The deep learning models presented in Section 4 detect the DGA domains from the embedded domain names. In this study, we solved the difficult multiclass classification problem of discriminating DGA types, not a simple binary classification. In particular, we compared four deep learning models: the two existing CNN and LSTM with Attention models (Sections 4.1 and 4.2, respectively) and the two proposed BiLSTM with Attention and CNN + BiLSTM ensemble models (Sections 4.3 and 4.4, respectively).

We implemented all models using *Python* under the Keras framework [25]. The loss function uses categorical cross entropy for multiclass classification. Table 3 lists the hyperparameters used for training the deep learning models. We selected the optimal values for these hyperparameters through repetitive experiments. We configured the epoch number to 10, learning rate to 1e-4, and batch size to 64. We also used the Adam optimizer as an optimization function.

5.4. Output. Finally, we classified the DGA classes through a fully connected layer. The number of outputs from the fully connected layer was equal to the number of DGA classes to be classified. We used the softmax function, which is widely used in multiclass classification problems as an activation function, i.e., after estimating the probabilities of the classes, we determined the maximum as the final DGA class for a domain name.

6. Experimental Evaluation

This section reports experiments on the proposed deep learning models using a realistic DGA dataset. Section 6.1 describes the experimental dataset and environment, and Section 6.2 presents the experimental results for the deep learning models.

6.1. Experimental Dataset and Environment. The dataset used in the experiment consisted of data from both DGA and non-DGA datasets. First, we used the DGA Domain Feed dataset provided by Bambenek Consulting OSINT [26] as the DGA domain data. This dataset contains more than 50 DGA types, including Banjori and Tinba. Among them, we used the top-20 DGA types in terms of volume. Next, we collected non-DGA data from Alexa Top Sites [27], which provides a list of popular website rankings. We assumed that the domains on the Alexa list are nonmalicious and trusted non-DGA domains. All data were collected on May 4, 2020. Table 4 describes the distribution of the experimental dataset, which consisted of 603,387 non-DGA and 832,271 DGA data.

We used a cross-validation technique to fairly evaluate the models; we divided the dataset into training, validation, and test data in an 8 : 1:1 ratio. More specifically, the experimental dataset contained 1,149,965 data for training the deep learning models, 142,126 validation data for measuring the performance of the models in the training process, and 143,567 test data for evaluating performance.

The experimental environment consisted of an Intel® Core™ i7-9700K CPU 3.60 GHz, 64 GB RAM, and a GeForce RTX 2080 8 GB GPU running an Ubuntu 16.04 LTS operating system with Anaconda 4.7.10 for development and evaluation.

In this study, we measured the performance of the models using precision, recall, and F1-score. First, precision represents the ratio of the actual true data among the data classified as true by a model. Second, recall represents the ratio of the data predicted by a model to be true to the actual true data. Third, the F1-score is calculated as a harmonic mean of the precision and recall. F1-score is popularly used as a general measure to accurately evaluate the performance of a model when the data are unbalanced, which was the case for this experiment, as can be seen from Table 4.

We also compared the performances of the deep learning models with two averages: *weighted-average* and *macro-average*. The weighted-average calculates the performance considering the total number of data. We calculated it by a weighted averaging of the number of samples classified in each class, which allows us to investigate the performance on the entire dataset. We obtained the macroaverage by independently calculating and summing the performances of each class and dividing them by the total number of classes. The macroaverage treats all classes as having the same number of data; thus, we can measure the average

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

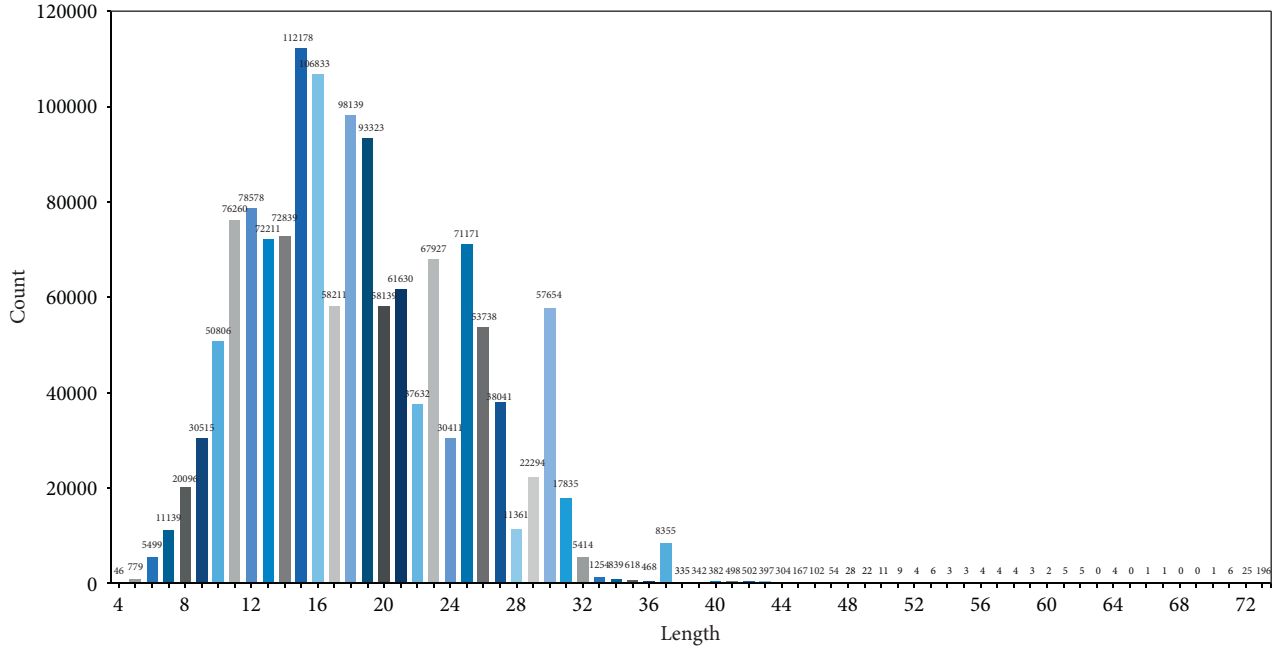

Figure 9: Length distribution of domain names.

Table 3: Hyperparameters of the deep learning models.

Parameter	Setting
Epoch	10
Batch size	64
Learning rate	$1e-4$
Optimizer	Adam

Table 4: Distribution of experimental dataset.

Class	DGA	Count
0	Non-DGA	603,387
1	Banjori	439,223
2	Tinba	66,789
3	Post	66,000
4	Ramnit	64,605
5	Qakbot	40,000
6	Necurs	32,768
7	Murofet	28,520
8	Shiotob/Urlzone/Bebloh	15,021
9	Simda	14,755
10	Ranbyus	13,200
11	Pykspa	10,019
12	Dyre	7,998
13	Kraken	7,878
14	Cryptolocker	6,000
15	Nymaim	6,000
16	Locky	4,014
17	Vawtrak	3,150
18	Shifu	2,331
19	Ramdo	2,000
20	P2P	2,000
DGA total	832,271	
Total (non-DGA + DGA)	1,435,658	

performance per class. If the amount of data is unbalanced in multiclass classification, the weighted-average is considered to be fairer than the macroaverage. However, to measure the performance for the overall data and the general performance for all classes, we compared both the weighted-average and macroaverage results.

6.2. Experimental Results. In this section, we have shown the experimental results of the DGA domain detection methods based on the two averaging methods described above. Section 6.2.1 covers evaluation of the overall classification performance using the weighted-average method. Section 6.2.2 covers evaluation of the classification performance for each class using the macroaverage method.

6.2.1. Overall Classification Performance. In this section, we compare the detection models through the weighted-average to evaluate their performances on the entire dataset. Table 5 and Figure 10 show the overall classification performance of the four deep learning models. The experiment was repeated five times, and each repeat was summed through the weighted-average for each performance measure (i.e., precision, recall, and F1-score).

Experimental results show that the F1-score of the CNN model was 0.9384, which is significantly lower than those of the LSTM models, which were 0.9597 or higher. This means that the LSTM models performed better than the CNN model for sequential data. Next, the F1-score of the BiLSTM with Attention model was 0.9618, which is higher than that of the LSTM with Attention model at 0.9597. This indicates that the bidirectional scheme of BiLSTM is more effective than the unidirectional scheme of LSTM. Finally, the CNN + BiLSTM ensemble model had the highest

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 5: Overall classification performances of deep learning models.

Run	CNN			LSTM_Attention			BiLSTM_Attention			Ensemble		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.9327	0.9383	0.9303	0.9598	0.9615	0.9591	0.9620	0.9636	0.9619	0.9670	0.9681	0.9664
2	0.9429	0.9487	0.9423	0.9607	0.9618	0.9604	0.9624	0.9635	0.9618	0.9673	0.9682	0.9664
3	0.9416	0.9475	0.9410	0.9598	0.9607	0.9590	0.9626	0.9634	0.9619	0.9677	0.9683	0.9668
4	0.9373	0.9461	0.9393	0.9605	0.9620	0.9597	0.9620	0.9636	0.9616	0.9677	0.9684	0.9666
5	0.9400	0.9462	0.9392	0.9606	0.9622	0.9604	0.9622	0.9630	0.9619	0.9680	0.9679	0.9670
Min	0.9327	0.9383	0.9303	0.9598	0.9607	0.9590	0.9620	0.9630	0.9616	0.9670	0.9679	0.9664
Max	0.9429	0.9487	0.9423	0.9607	0.9622	0.9604	0.9626	0.9636	0.9619	0.9680	0.9684	0.9670
Avg	0.9389	0.9454	0.9384	0.9603	0.9616	0.9597	0.9622	0.9634	0.9618	**0.9676**	**0.9682**	**0.9666**

*P : Precision, R : Recall, and F1 : F1-score.

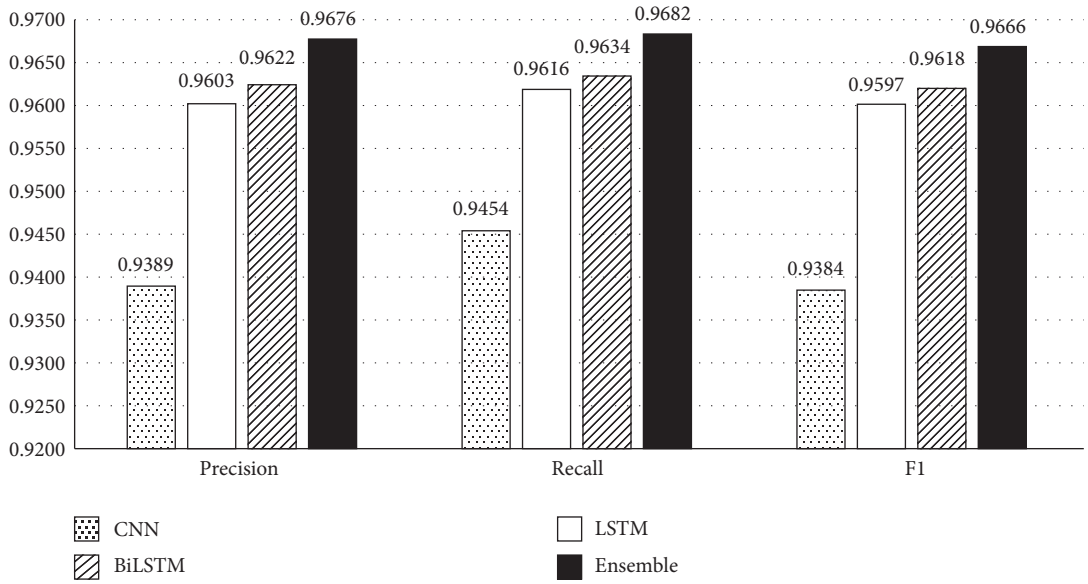

Figure 10: Overall performance comparison of deep learning models for DGA domain detection.

performance among all models with an F1-score of 0.9666. This excellent result indicates that the ensemble model fully exploits the advantages of both the CNN and BiLSTM models.

6.2.2. Class Classification Performance. We compared the classification performances of the deep learning models for each class. Table 6 lists the performance results for each class and their performances under the macroaverage and weighted-average measures. First, the proposed ensemble model showed the best F1-score for all classes except Post, Cryptolocker, and Locky. Even on the Post, Cryptolocker, and Locky classes, the ensemble model showed very high performance close to 1 or the best performance. These results indicate that the proposed ensemble model has the best performance for DGA domain classification, and we believe that the ensemble model learns with the advantages of both the CNN and BiLSTM models, leading to significant performance improvements. In particular, in the case of the Vawtrak class, the CNN model could not predict at all with an F1-score of 0.0000, and the BiLSTM model also showed a relatively low performance with an F1-score of 0.3918, but the CNN + BiLSTM ensemble model showed an almost two-fold improvement or better with an F1-score of 0.8401.

As described above, the ensemble model showed high performance for most classes by learning based on the advantages of the individual candidate models. We next investigate the general performances of the models by combining the performances on each class into the macroaverage. The F1-score macroaverage of the CNN model was the lowest at 0.6696, while that of the LSTM model was 0.7909. The BiLSTM model showed an F1-score of 0.8045, which is an improvement over the LSTM model of 0.0136, while the ensemble model also showed an F1-score of 0.8369, which is an improvement of 0.0460. We can interpret this as the proposed ensemble model generally predicting better for all classes than other deep learning models.

To compare the predictive performances in more detail, we present confusion matrices for the existing LSTM with Attention model and the proposed CNN + BiLSTM ensemble model in Figure 11. In these confusion matrices, the rows represent the actual classes, the columns represent the classes predicted by the deep learning model, and each block is gives the ratio in the range [0, 1] by normalizing the number of actual values and the number of predicted values.

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 6: Overall classification performances of deep learning models.

DGA	CNN			LSTM_Attention			BiLSTM_Attention			Ensemble			Support
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
Non-DGA (alexa)	0.9664	0.9978	0.9818	0.9827	0.9908	0.9867	0.9840	0.9910	0.9875	0.9869	0.9968	**0.9918**	60,424
Banjori	0.9995	1.0000	0.9997	0.9995	1.0000	0.9997	0.9995	1.0000	0.9998	0.9998	1.0000	**0.9999**	44,025
Tinba	0.8954	0.9862	0.9386	0.9294	0.9835	0.9557	0.9287	0.9890	0.9579	0.9273	0.9960	**0.9604**	6,720
Post	0.9998	0.9977	0.9988	1.0000	0.9998	**0.9999**	1.0000	0.9997	0.9998	1.0000	0.9997	0.9998	6,492
Ramnit	0.7637	0.8408	0.8004	0.8287	0.8964	0.8612	0.8321	0.8951	0.8625	0.8486	0.9023	**0.8747**	6,369
Qakbot	0.7164	0.6620	0.6882	0.7491	0.7995	0.7735	0.7676	0.7798	0.7737	0.7820	0.7970	**0.7894**	4,015
Necurs	0.7421	0.6927	0.7166	0.9310	0.7691	0.8424	0.9472	0.7953	0.8646	0.9483	0.8183	**0.8785**	3,248
Murofet	0.8158	0.7685	0.7914	0.8342	0.8286	0.8314	0.8230	0.8325	0.8277	0.8464	0.8209	**0.8335**	2,859
Shiotob/urlzone/bebloh	0.9817	0.8472	0.9095	0.9723	0.9136	0.9420	0.9858	0.9068	0.9447	0.9918	0.9136	**0.9511**	1,459
Simda	0.9399	0.9710	0.9552	0.9603	0.9791	0.9696	0.9586	0.9858	0.9720	0.9728	0.9912	**0.9819**	1,481
Ranbyus	0.5685	0.3792	0.4549	0.8054	0.8346	0.8197	0.8718	0.8323	0.8516	0.8536	0.8542	**0.8539**	1,324
Pykspa	0.9399	0.8992	0.9191	0.9336	0.9778	0.9552	0.9308	0.9899	0.9595	0.9790	0.9889	**0.9840**	992
Dyre	0.9910	1.0000	0.9955	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	769
Kraken	0.9843	0.7884	0.8755	0.9410	0.8438	0.8898	0.9329	0.8577	0.8937	0.9443	0.8539	**0.8968**	794
Cryptolocker	0.7212	0.1181	0.2030	0.5411	0.3732	0.4418	0.5152	0.4535	**0.4824**	0.6347	0.3228	0.4280	635
Nymaim	0.4000	0.0407	0.0740	0.6000	0.1885	0.2868	0.5377	0.2784	0.3669	0.6818	0.2547	**0.3708**	589
Locky	1.0000	0.0207	0.0405	0.7933	0.2736	0.4068	0.8696	0.3218	**0.4698**	0.9774	0.2989	0.4577	435
Vawtrak	0.0000	0.0000	0.0000	0.6916	0.2334	0.3491	0.7049	0.2713	0.3918	0.9484	0.7539	**0.8401**	317
Shifu	0.4945	0.7904	0.6084	0.8435	0.9651	0.9002	0.8550	0.9782	0.9124	0.8840	0.9651	**0.9228**	229
Ramdo	0.9846	0.9948	0.9897	0.9847	1.0000	0.9923	0.9847	1.0000	0.9923	0.9897	1.0000	**0.9948**	193
P2P	0.4118	0.0707	0.1207	0.4149	0.3939	0.4041	0.3957	0.3737	0.3844	0.4615	0.7273	**0.5647**	198
Macro-average	0.7770	0.6603	0.6696	0.8446	0.7735	0.7909	0.8488	0.7872	0.8045	0.8885	0.8217	**0.8369**	143,567
Weighted-average	0.9389	0.9454	0.9384	0.9603	0.9616	0.9597	0.9622	0.9634	0.9618	0.9676	0.9682	**0.9666**	143,567

*P : Precision, R : Recall, and F1 : F1-score.

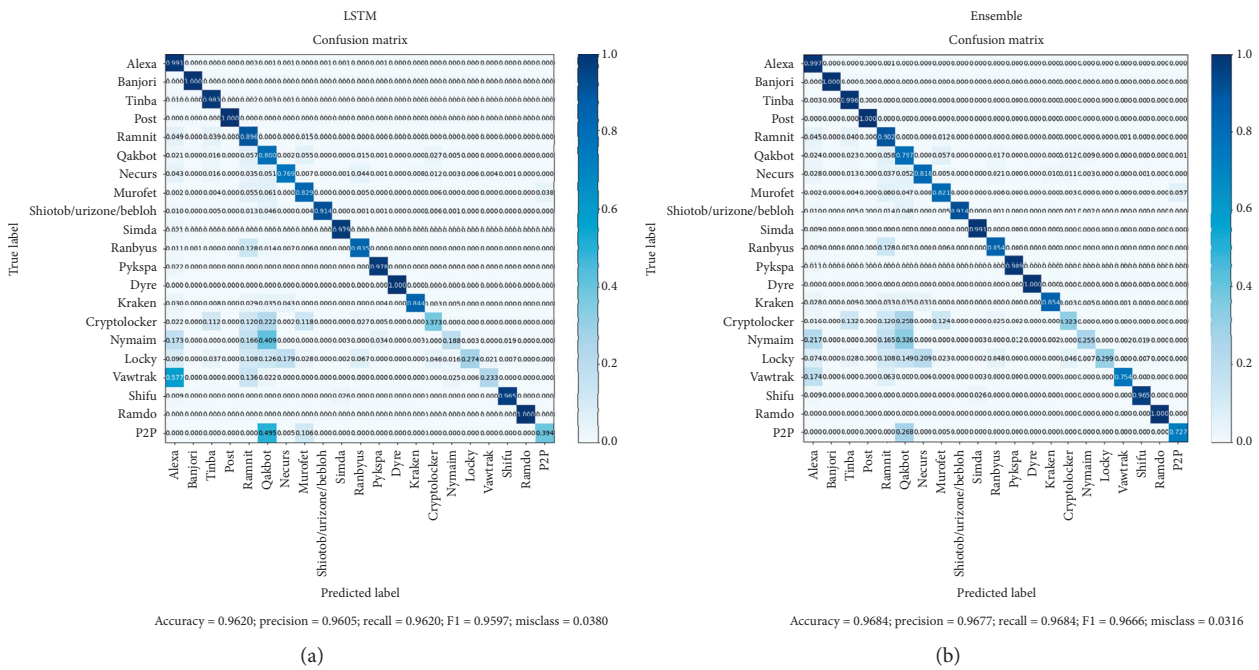

Figure 11: Confusion matrices of LSTM with Attention and CNN + BiLSTM ensemble models. (a) (a) LSTM with Attention model. (b) CNN + BiLSTM ensemble model.

The closer a block's value is to 1, the darker the color, and vice versa. Intuitively speaking, the ensemble model in Figure 11(b) has more dark blocks located on the diagonal compared to the LSTM model in Figure 11(a). This is because the ensemble model predicts more accurately than the LSTM model for all classes. In particular, the ensemble model predicts well even for the classes with little data. For example, in the case of the Vawtrak class in Figure 11(a), the LSTM with Attention model correctly predicts with a ratio of only 0.2330, but it incorrectly predicts non-DGA at a ratio of

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

0.5770, resulting in a higher false positive rate. In contrast, as shown in Figure 11(b), for the Vawtrak class, the ensemble model accurately predicts at a rate of 0.7540, and the false positive rate is 0.1740, which is significantly lower than that of the LSTM model. To summarize the accuracy in terms of the confusion matrices, the LSTM with Attention model achieved 0.9618, and the CNN + BiLSTM ensemble model achieved 0.9684, so we can also visually confirm the superiority of the proposed ensemble model.

7. Conclusions

In this study, we analyzed deep learning-based DGA domain detection methods and proposed two novel methods that significantly improve detection performance. In the existing DGA domain detection field, many works have mainly used the LSTM model, which has the disadvantage of learning only unidirectional information. To address this problem, we first proposed a BiLSTM-based DGA domain detection method that exploits bidirectional information. Next, we proposed an ensemble model-based method that learns the advantages of a CNN, which focuses on local sequence information, and BiLSTM, which focuses on global sequential information. Experimental evaluations using realistic domain data showed that the LSTM model had the lowest F1-score of 0.9597; the BiLSTM model gave an improved F1-score of 0.9618, while that of the ensemble model was 0.9666, which is the highest among the deep learning models. In addition, through detailed analysis, we confirmed that the ensemble model predicted with the highest performance for most classes. Therefore, we believe that the proposed BiLSTM with Attention and ensemble models are excellent approaches for classifying DGA domains in real time with higher accuracy than existing deep learning models.

With the continuous development of deep learning technologies, several new models have been introduced. In particular, the bidirectional gated recurrent unit (BiGRU), developed for a purpose similar to that of our models, is being introduced into text classification technology [28]. Similar to BiLSTM, BiGRU extends GRU, and it shows better performance than BiLSTM. Thus, we plan to study a BiGRU-based DGA domain detection method in the future. In addition, many DGAs are still emerging, and they threaten the existing DGA domain detection methods. John et al. [29] explained various adversarial attacks in malware detection classifiers, and suggested various steps and research directions to prevent them. Peck et al. [30] and Sidi et al. [31] presented novel DGA techniques that avoid DGA classifiers. Further, Corley et al. [32] proposed DomainGAN, which creates DGA domains with generative adversarial networks (GANs) that effectively generate data through deep learning. These are new types of DGA not covered in this paper. Therefore, we plan to study a robust DGA domain detection method that correctly detects new DGA types as well. Finally, we plan to study the malicious URL detection problem by expanding the application domain in the DGA classification which is mainly considered in this paper.

Data Availability

All the data files used in the experiments are available at https://github.com/Juhong-Namgung/Malicious-URL-and-DGA-Domain-Detection-using-Deep-Learning.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was partly supported by Korea Electric Power Corporation (Grant number: R18XA05). This research was also partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1085311).

References

[1] B. Stone-Gross, M. Cova, L. Cavallaro, and B. Gilbert, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 635–647, Chicago, IL, USA, November 2009.

[2] M. Kührer, C. Rossow, and T. Holz, "Paint it black: evaluating the effectiveness of malware blacklists," in *Proceedings of the 17th Int'l Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 1–21, Gothenburg, Sweden, September 2014.

[3] Y. Zhang, Y. Zhang, and J. Xiao, "Detecting the DGA-based malicious domain names," in *Proceedings of the Int'l Conference on Trustworthy Computing and Services*, pp. 130–137, Heidelberg, Berlin, November 2013.

[4] X. Luo, L. Wang, Z. Yang, M. Sun, and J. Wang, "DGASensor: fast detection for DGA-based malwares," in *Proceedings of the 5th Int'l Conference on Communications and Broadband Networking*, pp. 47–53, Bail, Indonesia, February 2017.

[5] A. J. Ferrante, "The impact of GDPR on WHOIS: implications for businesses facing cybercrime," *Cyber Security: A Peer-Reviewed Journal*, vol. 2, no. 2, pp. 143–148, 2018.

[6] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol. 275, pp. 2401–2413, 2018.

[7] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," in *Proceedings of the 14th Int'l Conference on Availability, Reliability, and Security*, pp. 1–10, Canterbury, UK, August 2019.

[8] Z. Feng, C. Shuo, and W. Xiaochuan, "Classification for DGA-based malicious domain names with deep learning architectures," *Int'l Journal of Intelligent Information Systems*, vol. 6, no. 6, pp. 67–71, 2017.

[9] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. D. Cock, "Character level based detection of DGA domain names," in *Proceedings of the Int'l Joint Conference on Neural Networks*, pp. 1–8, Rio de Janeiro, Brazil, July 2018.

[10] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," 2016, https://arxiv.org/abs/1611.00791.

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

[11] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, and H. Wu, "DGA domain name classification method based on long short-term memory with attention mechanism," *Applied Sciences*, vol. 9, no. 20, 2019.

[12] B. Yu, D. Gray, J. Pan, M. D. Cock, and A. C. Nascimento, "Inline DGA detection with deep networks," in *Proceedings of the 2017 IEEE Int'l Conference on Data Mining Workshops*, pp. 683–692, New Orleans, LO, USA, November 2017.

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[14] T. G. Dietterich, "Ensemble methods in machine learning,," in *Proceedings of the Int'l Workshop on Multiple Classifier Systems*, pp. 1–15, Berlin, Heidelberg, December 2000.

[15] S. Yadav, A. K. K. Reddy, and A. L. N. Reddy, "Detecting algorithmically generated malicious domain names,," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 48–61, New York, NY, USA, November 2010.

[16] Z. Wei-wei, G. Jian, and L. Qian, "Detecting machine generated domain names based on morpheme features," in *Proceedings of the 1st Int'l Workshop on Cloud Computing and Information Security*, pp. 408–411, Shanghai, China, November 2013.

[17] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Proceedings of the Int'l Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 192–211, London, UK, July 2014.

[18] M. Antonakakis, R. Perdisci, Y. Nadji et al., "From throw-away traffic to bots: detecting the rise of DGA-based malware," in *Proceedings of the 21st USENIX Security Symposium*, pp. 491–506, Washington, DC, USA, August 2012.

[19] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[20] Banking Trojan Banjori Analysis Report, https://nsfocusglobal.com/banking-trojan-banjori-analysis-report/.

[21] J. Stiborek, T. Pevný, and M. Rehák, "Probabilistic analysis of dynamic malware traces," *Computers & Security*, vol. 74, pp. 221–239, 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 34, 2017.

[23] J. Saxe and K. Berlin, "expose: a character-level convolutional neural network with embeddings for detecting malicious urls," *File Paths, and Registry Keys*, vol. 34, 2017.

[24] D. Opitz and R. Maclin, "Popular ensemble methods: an empirical study," *Journal of Artificial Intelligence Research*, vol. 11, no. 1, pp. 169–198, 1999.

[25] Keras, http://keras.io/.

[26] Bambenek Consulting OSINT, http://osint.bambenekconsulting.com/feed/.

[27] Alexa Top Sites, http://alexa.com/topsites/.

[28] J. Zhang, F. Liu, W. Xu, and H. Yu, "Feature fusion text classification model combining CNN and BiGRU with multi-attention mechanism," *Future Internet*, vol. 11, no. 11, pp. 1–24, 2019.

[29] T. S. John and T. Thomas, "Adversarial attacks and defenses in malware detection classifiers," *Handbook of Research on Cloud Computing and Big Data Applications In IoT, IGI Global*, vol. 34, pp. 127–150, 2019.

[30] J. Peck, C. Nie, R. Sivaguru et al., "Charbot: a simple and effective method for evading dga classifiers," *IEEE Access*, vol. 7, 2019.

[31] L. Sidi, A. Nadler, and A. Shabtai, "Maskdga: a black-box evasion technique against dga classifiers and adversarial defenses," 2019, https://arxiv.org/abs/1902.08909.

[32] I. Corley, J. Lwowski, and J. Hoffman, "Domaingan: generating adversarial examples to attack domain generation algorithm classifiers," 2020, https://arxiv.org/abs/1911.06285.

2037, 2021, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2021/8887881, Wiley Online Library on [13/02/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License