

Deep Convolutional Neural Network and Character Level Embedding for DGA Detection

João Rafael Gregório¹^a, Adriano Mauro Cansian¹^b, Leandro Alves Neves¹^c
and Denis Henrique Pinheiro Salvadeo²^d

¹*Department of Computer Science and Statistics (DCCE), São Paulo State University (UNESP),
São José do Rio Preto, São Paulo, Brazil*

²*Institute of Geosciences and Exact Sciences (IGCE), São Paulo State University (UNESP), Rio Claro, Brazil*

Keywords: Domain Generation Algorithms, DGA, Convolutional Neural Networks, Embedding, NLP, Short Text Classification, Cybersecurity.

Abstract: Domain generation algorithms (DGA) are algorithms that generate domain names commonly used by botnets and malware to maintain and obfuscate communication between a botclient and command and control (C2) servers. In this work, a method is proposed to detect DGAs based on the classification of short texts, highlighting the use of character-level embedding in the neural network input to obtain meta-features related to the morphology of domain names. A convolutional neural network structure has been used to extract new meta-features from the vectors provided by the embedding layer. Furthermore, relu layers have been used to zero out all non-positive values, and maxpooling layers to analyze specific parts of the obtained meta-features. The tests have been carried out using the Majestic Million dataset for examples of legitimate domains and the NetLab360 dataset for examples of DGA domains, composed of around 56 DGA families. The results obtained have an average accuracy of 99.12% and a precision rate of 99.33%. This work contributes with a natural language processing (NLP) approach to DGA detection, presents the impact of using character-level embedding, relu and maxpooling on the results obtained, and a DGA detection model based on deep neural networks, without feature engineering, with competitive metrics.

1 INTRODUCTION


Botnets are groups of computers or devices infected by malware that can be controlled remotely through a command and control (C2) server (Kambourakis et al., 2019). Attackers use botnets to carry out various types of cyberattacks, such as distributed denial of service (DDoS) attacks, directing members of their botnet to attack a specific target or for the systematic leakage of information from corporate and government networks (Shahzad et al., 2021).


Domain generation algorithms (DGA) are existing codes in most malware families that generate domain names. The generated domains are used to maintain communication between the infected machine and the C2 servers (Wong, 2023; Huang et al., 2022). This strategy allows the C2s to, for example, change their


IP addresses without affecting communication with botnet member machines. Therefore, DGAs are used to increase the communication resilience between C2 and botnet members and improve the obfuscation capacity of the C2 server (Kambourakis et al., 2019).


Attackers register large numbers of domains compatible with their generating algorithms on registrants¹ on the internet. This strategy allows that in the event of a domain being listed due to abuse, another domain name can be used. This large number of DGA domain names makes using blocklists a problematic alternative, as a list of a few thousand domains will likely be a performance drag on any firewall system (Ren et al., 2020). Thus, identifying requests for DGA domains on a computer network, or even noticing them during the registration process, can help mitigate botnet-related threats, prevent impending attacks on corporate networks, and support

¹Companies authorized to sell second-level domain names directly to individuals or companies (Tanenbaum et al., 2021).

^a <https://orcid.org/0000-0001-7783-2567>

^b <https://orcid.org/0000-0003-4494-1454>

^c <https://orcid.org/0000-0001-8580-7054>

^d <https://orcid.org/0000-0001-8942-0033>

proactive cybersecurity.

In this context, some works focused on detecting DGAs have been developed using approaches based on classic machine learning, with the manual development of attributes, to map the characteristics that characterize these domain names. For instance, the proposal of (Piras et al., 2022) presents an explainable approach based on attributes of data collected from domain name system (DNS) servers and compares the benchmarked results of some classical machine learning algorithms such as random forest, adaboost and decision tree. That approach depends on data collection windows related to DGAs on DNS servers. The work presents the comparison of results obtained with windows of five and fourteen days, demonstrating that the best results are obtained with the largest data capture window.

Approaches using deep neural networks (DNN) and lexical features of DGAs are also found in recent literature. B. Yu, J. Pan, J. Hu, A. Nascimento and M. De Cock (Yu et al., 2018) discuss some models of DNNs for the detection of DGAs comparing them to results from models based on lexical features. The researchers applied models of convolutional neural networks (CNN) and long-short term memory (LSTM) trained with about 1.6 million samples (80% of the dataset) and obtained results of about 98% accuracy. Vranken, H.; Alizadeh, H. (Vranken and Alizadeh, 2022) propose the use of Term Frequency-Inverse Document Frequency (TF-IDF) to evaluate the relevance of 5000 n-grams of 1, 2, and 3 letters in the composition of domain names. Their model based on an LSTM network achieved around 97% accuracy. The approach with manually developed characteristics can be more complex as it requires more specialized work in addition to possibly failing to include specific characteristics of DGA families that may not be present in the used data set.

Other approaches use DNNs, directly exploring domain names as data input, without manually producing characteristics. Ren, F., Jiang, Z., Wang, X. and Liu, J. (Ren et al., 2020) present a hybrid model of a DNN, with CNN, bi-directional long-short term memory (BiLSTM) layers and an attention mechanism, also using 80% of the dataset for training and 20% for testing, about 1.3 million samples. The obtained results have been of the order of 99% accuracy for the classification between DGA and legitimate. The same work also presents a classification by DGA family, using 24 families of DGA domains. H. Shahzad, A. R. Sattar and J. Skandaraniyam (Shahzad et al., 2021) propose removing the top-level domain (TLD) during pre-processing, using “google” instead of “google.com”. The proposed DNN model has been

based on three types of recurrent networks (LSTM, Bi-LSTM, and GRU-Gated Recurrent Unit). The results have been about 87% accuracy with a training dataset of about 1.8 million samples and 44 DGA families. It is observed that recent works that address the detection of DGAs with natural language processing (NLP) techniques also use manually developed resources at some point.

The works that propose directly using domain names as input for their models are based on hybrid structures, normally using CNN, Recurrent, LSTM, BiLSTM networks, or attention mechanisms in the same model. Yu Wang, Rui Pan, Zuchao Wang and Lingqi Li (Wang et al., 2023) utilize the dataset provided by NetLab360 for DGA examples, which includes approximately 35 DGA families. They encode the input characters into values ranging from 0 to 38 to obtain the initial feature vector. A CNN model concatenated with a BiLSTM network is presented to extract meta-features for the purpose of detecting DGAs. The results reported in the study show an accuracy of 94.51%, a recall of 95.05%, and a precision of 93.11% for binary detection. Weiqing Huang, Yangyang Zong, Zhixin Shi, Leiqi Wang and Pengcheng Liu (Huang et al., 2022) uses pre-trained word embedding models such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018) to generate the initial feature vector from domain names. This initial vector is fed into a CNN and then to the classifier, achieving an accuracy of 96.08% in its best result.

Thus, an approach based on NLP with character-level embedding and CNN, exclusively using domain names as input for the proposed model, to extract the best meta-features for detecting DGAs has not yet been completely covered in specialized literature. Therefore, our method aims to contribute by verifying the effectiveness of using character-level embedding in detecting DGAs in datasets with heterogeneity of DGA families.

In this scenario, this work presents an approach based on a CNN, to extract features from the domain names themselves, without the feature engineering process. The proposal also used character-level embedding at the proposed network’s input to find characteristics related to the morphology of the analyzed domain names. Furthermore, the proposed approach uses relu layers to make the network sparse, returning zero for any non-positive value, and maxpooling layers to highlight specific characteristics of the analyzed classes. The main contributions of this paper are:

- Indicate the use of character-level embedding as an effective strategy to improve DGA detection results in DNN architectures.

- Describe how the combined use of relu, maxpooling and embedding layers can improve the results and performance of CNNs in detecting DGAs.
- Present a DNN architecture with competitive results in detecting DGAs without feature engineering.

This paper is organized as follows. Section 2 presents the methodology used in data pre-processing, the datasets used, tools, and the technologies that supported the development of this work, in addition to the description of the proposed model. Section 3 presents the results obtained in distinct tests, comparisons with different combinations of the proposed model and an overview of the obtained results in relation to the literature. Finally, in Section 4, conclusions and proposals for future work are presented.

2 METHODOLOGY

This Section presents the methodology used for the development of the proposed model, pre-processing of datasets, and the technologies used. Section 2.1 presents the development environment and the technologies applied, the data sets, their pre-processing, and division into sets for training and testing. Section 2.2 presents the proposed model, each of the layers used along with their functions, hyperparameters, and a summary to facilitate the reproduction of this experiment. Section 2.3 presents the metrics used to evaluate the results obtained with training and testing the proposed model.

2.1 Development Environment and Datasets

The algorithms of this work have been executed using the Kaggle Notebook platform (Kaggle Team, 2023), a public and free platform, where it is possible to use a monthly Graphics Processing Unit (GPU) processing quota, essential for processing large volumes of data and training DNNs in less time. All codes have been developed in the Python 3 programming language, using the Tensorflow (Abadi et al., 2015), Pandas (The Pandas Development Team, 2020), and Numpy (Harris et al., 2020) libraries. The notebook containing the complete experiment has been published, thus allowing the complete reproduction of this work. The source code is available at <https://www.kaggle.com/code/rafaelgregorio/deep-convolutional-embedding-dga-detector>.

Also, the Majestic Top Million (Majestic, 2023) dataset containing 1 million domains has been used

as the legitimate domain data source. As a source of DGA domain names, the dataset provided by NetLab360 (NetLab 360, 2022) has been used in this investigation, containing about 1 million DGA domains, distributed across 56 families, collected in real traffic. Legitimate domain names have been labeled 0, and the domain names from DGA families have been labeled 1.

Domain names have been encoded character by character to ASCII code. The maximum length checked across the domain name samples has been 73 characters, so smaller domain names have been padded with zeros from the end to the maximum length. Table 1 shows examples of domain names already encoded to ASCII.

Table 1: Examples of domain names and each character encoded by ASCII.

domain	label	1	2	...
gafsjfdues.com	1	103	97	...
yvhbrwh.info	1	121	118	...
nwqpswrwsn.us	1	110	119	...
awnspwrpnn.com	1	97	119	...
cghgfdguvol.net	1	99	103	...
mayermoney.com	0	109	97	...
sqlservercentral.com	0	115	113	...
lottomatica.it	0	108	111	...

In order to train the proposed model, 20% of the total data have been proportionally selected. To test, five samples, each one with 5% of the total data have been selected randomly and proportionally.

2.2 Proposed Model

The model proposed in this work has been constructed based on an approach rooted in a DNN, using domain names converted, character by character, to their respective ASCII codes as input data. This initial feature vector is then passed through an embedding layer that converts each ASCII value into a vector of 20 features related to the context of the character in the initial string. The output of the embedding layer is connected to a CNN, which aims to extract relevant features from the domain names based on the vectors received from the upper embedding layer. In addition, the relu layers have been applied to sparsify the network, returning zero for any non-positive values, and maxpooling layers to highlight specific features of the analyzed classes.

2.2.1 Embedding Layer

Embedding layers are widely used in NLP, where each word is represented by an array of floating point values (Goodfellow et al., 2016; Zhang et al., 2023). Embedding layers are trainable. Therefore, the representation of each word can be learned by the network. In this work, embedding at the character level has been used. Therefore, each character of a domain name, after being transposed into its ASCII code, is taken to the embedding layer, where it is represented by a vector of 20 values. This feature vector constructed by the embedding layer is then passed to the CNN part of the proposed model.

2.2.2 Deep Convolutional Neural Network

Convolutional neural networks are capable of finding specific features in images, requiring minimal pre-processing (Goodfellow et al., 2016; Zhang et al., 2023). In this work, the CNN part of the proposed model is designed to extract relevant features for the analyzed classes, legitimate and DGAs, from the feature vector generated by the embedding layer positioned at the model's input data. Our model has six convolutional layers of distinct dimensions, forming a deep CNN. The dimensions of each of the convolutional layers and also the other layers used in the proposed model are better illustrated in Figure 1.

2.2.3 ReLU and MaxPooling Layers

Positioned within the convolutional layers are relu and maxpooling layers, contributing to maximizing the model's intrinsic ability to discern and identify pertinent attributes associated with each distinct class (Bahera H. Nayef, 2023). Rectified linear unit (ReLU) is an activation function that, in practice, turns every negative input into zero (Zhang et al., 2023), as described by equation 1. This feature is desirable for two reasons, firstly neurons in the network that receive zero values will not be activated, making the network sparse. Secondly, the positive values that will be treated by the network tend to be more representative. Thus, the relu layer ends up working as a high-contrast, highlighting important areas.

$$ReLU(x) = (x)^+ = \max(0, x) \quad (1)$$

MaxPooling layers downsample the feature map extracted by the convolutional layers (Zhang et al., 2023). In the model proposed in this work, maxpooling layers are used so that the feature maps highlighted by the relu layer can be condensed in the maxpooling layer. This integration of layers and architectural elements is intended to enhance the model's ca-

capacity to extract and process information efficiently, thereby facilitating its performance in the identification of significant features relevant to the classification of diverse classes.

2.2.4 Flatten and Dense Layers

The flatten layer has been used after the convolutional part of the proposed model to reshape the format of the feature vector produced by the upper layers, making it compatible with the fully connected dense layers positioned at the end of the model. Finally, two fully connected layers, with 15 and 1 neuron respectively, act as classifiers using the feature vector received from the flatten layer. Figure 1 shows the summary of the model, its layers, and its dimensions.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 73, 20)	5120
conv1d (Conv1D)	(None, 73, 500)	70500
re_lu (ReLU)	(None, 73, 500)	0
conv1d_1 (Conv1D)	(None, 73, 250)	750250
re_lu_1 (ReLU)	(None, 73, 250)	0
max_pooling1d (MaxPooling1D)	(None, 73, 250)	0
conv1d_2 (Conv1D)	(None, 73, 120)	150120
re_lu_2 (ReLU)	(None, 73, 120)	0
conv1d_3 (Conv1D)	(None, 73, 60)	36060
re_lu_3 (ReLU)	(None, 73, 60)	0
max_pooling1d_1 (MaxPooling1D)	(None, 73, 60)	0
conv1d_4 (Conv1D)	(None, 73, 30)	7230
re_lu_4 (ReLU)	(None, 73, 30)	0
conv1d_5 (Conv1D)	(None, 73, 30)	2730
re_lu_5 (ReLU)	(None, 73, 30)	0
max_pooling1d_2 (MaxPooling1D)	(None, 73, 30)	0
flatten (Flatten)	(None, 2190)	0
dense (Dense)	(None, 15)	32865
dense_1 (Dense)	(None, 1)	16

Total params: 1,054,891
Trainable params: 1,054,891
Non-trainable params: 0

Figure 1: An illustrative overview of the proposed model.

2.2.5 Hyperparameters and the Experiment Execution

Another important aspect of the proposed model is the hyperparameters. In deep learning models, hyperparameters are parameters that are not learned during the model training but need to be defined before the start of this process (Koutsoukas et al., 2017; Dalli, 2022). They are set externally to the model and play a crucial role in the architecture and training of the neural network. Unlike model parameters such as weights and biases, which are adjusted automatically

by the optimization algorithm during training, hyperparameters are predetermined and directly impact the performance and behavior of the model. Thus, the hyperparameters used to compile the proposed model are shown in Table 2.

Table 2: Hyperparameters used to compile the proposed model.

Hyperparameter	Value
L1 Regularizer	1e-5
L2 Regularizer	1e-4
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4
Batch Size	500

The L1 and L2 regularizers have been used to apply penalties to the layer parameters, being added to the loss function to avoid overfitting (Salehin and Kang, 2023). The low values adopted, 1e-5 and 1e-4 respectively, are low enough not to have a sudden impact on the training process. Adam optimizer has been selected because it is a computationally efficient stochastic gradient descent method for problems with a large number of parameters and data (Kingma and Ba, 2017). ReLU activation function has been chosen to increase the sparsity of the network due to its characteristic of turning every negative value received into zero (Zhang et al., 2023) and the network’s learning rate has been set to a low value, 1e-4, to seek a slower evolution of parameter updates and thus seek the best global minimum loss.

The experiment has been conducted in three stages. The first involved data pre-processing, where domain names have been converted, character by character, to their respective ASCII codes, up to a limit of 73 characters—the maximum length observed in the examples used. Domain names shorter than 73 characters have been padded with zeros from the end. In a second phase, a subset of 20%, approximately 400 thousand examples, has been selected from the complete dataset for training the proposed model. In addition, five separate samples of 5% each have been set aside for conducting tests with the trained model. These datasets have been used to train and test, in addition to the main model, two other models with different layers combinations to assess the impact of using embedding, relu and maxpooling on the obtained results. The results are presented in Section 3. Finally, the model has been trained for 20 epochs and the best accuracy has been selected using the callback function. Then the prediction of the five test samples has been performed. For each test, an evaluation has been made based on the metrics presented in Subsec-

tion 2.3. The complete process diagram is detailed in Figure 2.

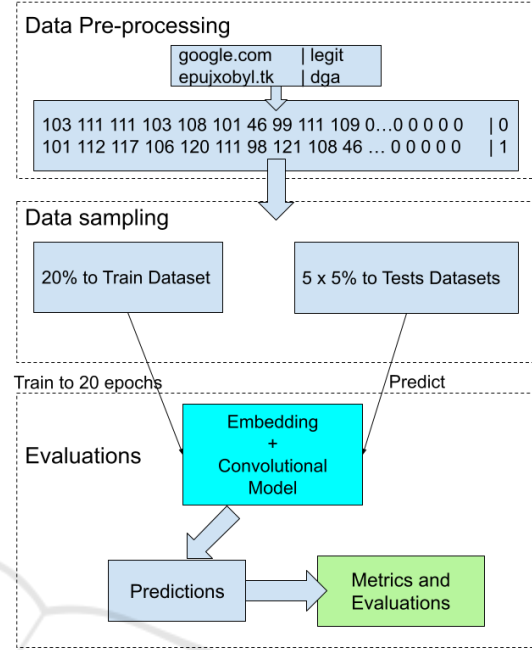


Figure 2: Experiment process diagram with data pre-processing, data sampling, training and testing steps.

2.3 Metrics and Evaluations

The model has been evaluated using the most commonly accepted metrics currently: Precision (equation 2), Recall (equation 3), Accuracy (equation 4), False Positive Rate (FPR) (equation 5). These metrics are obtained as follows:

$$Precision(Prec.) = \frac{TP}{FP + TP} \quad (2)$$

$$Recall(Rec.) = \frac{TP}{FN + TP} \quad (3)$$

$$Accuracy(Acc.) = \frac{TN + TP}{TN + TP + FN + FP} \quad (4)$$

$$FalsePositiveRate(FPR) = \frac{FP}{TN + FP} \quad (5)$$

Where:

- True Positive (TP): DGA domain names classified as DGA;
- True Negative (TN): Legitimate domain names classified as Legitimate;
- False Positive (FP): Legitimate domain names classified as DGA;
- False Negative (FN): DGA domain names classified as Legitimate;

3 RESULTS

This section presents the results obtained in the developed experiments. In order to evaluate the performance of the proposed model, called Model 1, and the effectiveness of combining the embedding, relu and maxpooling layers with the deep CNN for detecting DGAs, two other models have been trained and tested using the same datasets.

Model 2 has the same CNN and embedding network structure but without the relu and maxpooling layers. Model 3 has the same CNN structure with relu and maxpooling layers but without the embedding layer in the data input. The structures of the evaluated models are presented in Table 3.

Table 3: Tested models and their respective base architectures.

Model	Architecture
Model 1	CNN+Embedding+ReLU+MaxPooling
Model 2	CNN+Embedding
Model 3	CNN+ReLU+MaxPooling

The average (Avg.), median (Med.) and standard deviation (Stdev.) of the metrics results obtained for the positive class in the five test samples for each of the three models have been calculated to compare the results. Table 4 presents the results obtained by Model 1.

Table 4: Results of the metrics obtained by Model 1 on each of the five test data sets, their average, median, and standard deviation.

	Prec.	Rec.	Acc.	FPR
Test 1	99.32%	98.92%	99.11%	0.69%
Test 2	99.36%	98.98%	99.16%	0.65%
Test 3	99.33%	98.95%	99.13%	0.68%
Test 4	99.33%	98.94%	99.13%	0.68%
Test 5	99.31%	98.86%	99.08%	0.70%
Avg.	99.33%	98.93%	99.12%	0.68%
Med.	99.33%	98.94%	99.13%	0.68%
Stdev.	0.02%	0.04%	0.03%	0.02%

It is possible to verify that Model 1, complete with the embedding layer in the data input, relu and maxpooling layers, obtained satisfactory results in the tests carried out with the dataset used. It is noted that the FPR remained below 1% in all test sets and the average accuracy of 99.33% reinforces this assessment. The average recall of 98.93% demonstrates that the model has been indeed successful in capturing the majority of the DGA examples presented. The average accuracy of 99.12% again demonstrates the balance of the proposed model depending on both classes examined.

Table 5 presents the results obtained by Model 2.

Table 5: Results of the metrics obtained by Model 2 on each of the five test data sets, their average, median, and standard deviation.

	Prec.	Rec.	Acc.	FPR
Test 1	98.32%	97.69%	97.99%	1.70%
Test 2	98.28%	97.86%	98.06%	1.74%
Test 3	98.38%	97.75%	98.05%	1.64%
Test 4	98.37%	97.83%	98.09%	1.65%
Test 5	98.37%	97.69%	98.02%	1.65%
Avg.	98.34%	97.76%	98.04%	1.68%
Med.	98.37%	97.75%	98.05%	1.65%
Stdev.	0.04%	0.08%	0.04%	0.04%

Based on the results obtained by Model 2, a model without the relu and maxpooling layers, it is noticed that in all metrics the results have been slightly below those obtained by Model 1, around 1% lower. The FPR more than doubled, from 0.68% in Model 1 to 1.68% in Model 2. These results demonstrate the importance of the relu and maxpooling layers for detecting DGAs by the proposed model.

Table 6 presents the results obtained by Model 3.

Table 6: Results of the metrics obtained by Model 3 on each of the five test data sets, their average, median, and standard deviation.

	Prec.	Rec.	Acc.	FPR
Test 1	95.97%	91.42%	93.74%	3.91%
Test 2	95.93%	91.39%	93.70%	3.94%
Test 3	95.95%	91.33%	93.69%	3.92%
Test 4	95.90%	91.25%	93.62%	3.97%
Test 5	95.79%	91.24%	93.56%	4.08%
Avg.	95.91%	91.33%	93.66%	3.96%
Med.	95.93%	91.33%	93.69%	3.94%
Stdev.	0.07%	0.08%	0.07%	0.07%

The results obtained by Model 3, a model without the embedding layer in the data input, have been even worse than those of Model 2 in relation to Model 1. All metrics have been around 4% below the metrics obtained by Model 1 and the FPR increased about sixfold, from 0.68% in Model 1 to 3.96% in Model 3. These results confirm the importance of using the embedding layer in the data input of the proposed model. Table 7 compares the average of the metrics obtained by the three models.

Table 7: Comparison of the average metrics values obtained by the three tested models.

Model	Prec.	Rec.	Acc.	FPR
Model 1	99.33%	98.93%	99.12%	0.68%
Model 2	98.34%	97.76%	98.04%	1.68%
Model 3	95.91%	91.33%	93.66%	3.96%

Observing the average results of the metrics presented in Table 7, it can be verified that Model 1 pro-

posed with the combination of the different layers embedding, relu, and maxpooling provided the best results. It can also be verified that removing the embedding layer (Model 3) resulted in a greater impact on metrics than removing the relu and maxpooling layers (Model 2). Therefore, it is possible to state, based on these results, that the use of character-level embedding has been fundamental for achieving the best metrics. However, the relu and maxpooling layers also contribute to the better result obtained by Model 1, which is the model proposed in this work.

3.1 A Comprehensive Overview of the Proposed Model in Relation to the Literature

In specialized literature, distinct approaches have been proposed for detecting DGAs, exploring different datasets. The models have been developed based on diverse combinations, using both classical machine learning (ML) and deep learning (DL) techniques. Features have been extracted either manually or through meta-features obtained by neural network models. A comprehensive overview is crucial to highlight the quality of this study, which adopts a natural language processing approach using character-level embedding and CNN. This overview is presented in Table 8.

Table 8: Overview of the results of some recent works, with different approaches for detecting DGAs.

Work	Architecture	Acc.
Proposed Model 1	Embedding + CNN + ReLU + MaxPooling	99.13%
Ren et al., 2020	CNN + BiLSTM + Attention	98.82%
Huang et al., 2022	Pré-trained Embeddings + CNN + MaxPooling	96.08%
Wang et al., 2023	CNN + BiLSTM	94.51%
Vranken and Alizadeh, 2022	TF-IDF + LSTM	90.69%
Shahzad et al., 2021	LSTM + BiLSTM + GRU	87.00%

Analyzing the data presented in the overview, it is observed that the results obtained by this study align with works found in specialized literature, considering the variety of existing approaches. In this comprehensive overview, Model 1 exhibited higher accuracy than the other approaches. However, this naturally does not imply that our model is superior, given the differences in techniques and datasets. Nevertheless, the approach has demonstrated effectiveness and ro-

bust metrics with the dataset used. Finally, it is worth noting that the majority of works focused on DGA detection show results close to the ideal, with metrics above 90%. Therefore, the contribution of this work lies in presenting an approach and demonstrating the effectiveness of using character-level embedding for the DGA detection problem.

4 CONCLUSION AND FUTURE WORKS

This work presented a brief overview of DGA, how they represent important cyber threats today, and presents some recent works, their results for detecting DGAs, as well as their approaches.

The proposed model proved to be robust, with results aligned with the results of the most current studies in the same direction. From the tests carried out with changes to the main Model 1, it has been possible to verify the impact of the combined use of the relu and maxpooling layers for the CNN model in detecting DGAs. It has been observed from the results obtained that the inclusion of the embedding layer contributed significantly to improving the results obtained.

Checked that the impact of using the relu and maxpooling layers has been smaller in the results obtained than using the embedding layer at the character-level in the data input. While the model without the relu and maxpooling layers obtained metrics around 1% lower than the full model in the tests, the model without embedding obtained results around 4% lower. Thus, the complete model (Model 1), with all layers, has been more efficient in detecting DGAs during the experiment.

It is noteworthy that for the FPR metric, Model 1 achieved superior performance, with an FPR of less than half compared to Model 2 and almost six times lower than Model 3. In high DNS query volume environments, FPR can significantly impact the performance and usability of predictive models.

In future works, it is suggested an approach based on Transformers, a recent technique that has surpassed in some cases the results of CNNs. Studies focusing on specific DGA families may help to better understand this threat type, perhaps using models based on NLP approaches. It is also suggested an unsupervised or semi-supervised approach to detect DGA domains, including some integration with DNS traffic collection mechanisms in local networks, thus being able to work as a powerful cybersecurity tool.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), the National Council for Scientific and Technological Development CNPq (Grant #313643/2021-0) and Núcleo de Informação e Coordenação do Ponto BR - NIC.br.

REFERENCES

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- Bahera H. Nayef, Siti Norul Huda Sheikh Abdullah, R. S. A. M. S. (2023). Text extraction with optimal bi-lstm. *Computers, Materials & Continua*, 76(3):3549–3567.
- Dalli, A. (2022). Impact of hyperparameters on deep learning model for customer churn prediction in telecommunication sector. *Mathematical Problems in Engineering*, 2022:4720539.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.
- Huang, W., Zong, Y., Shi, Z., Wang, L., and Liu, P. (2022). Pepec: A deep parallel convolutional neural network model with pre-trained embeddings for dga detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.
- Kaggle Team (2023). Kaggle. <https://www.kaggle.com/>.
- Kambourakis, G., Anagnostopoulos, M., Meng, W., and Zhou, P. (2019). *Botnets: Architectures, Countermeasures, and Challenges*. CRC Press, Boca Raton, 1 edition.
- Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.
- Koutsoukas, A., Monaghan, K. J., Li, X., and Huan, J. (2017). Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics*, 9(1):42.
- Majestic (2023). Majestic Million. <https://pt.majestic.com/reports/majestic-million>.
- NetLab 360 (2022). NetLab360. <https://data.netlab.360.com/>.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.
- Piras, G., Pintor, M., Demetrio, L., and Biggio, B. (2022). Explaining machine learning dga detectors from dns traffic data.
- Ren, F., Jiang, Z., Wang, X., and Liu, J. (2020). A dga domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity*, 3:4.
- Salehin, I. and Kang, D.-K. (2023). A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics*, 12(14).
- Shahzad, H., Sattar, A., and Skandaraniyam, J. (2021). Dga domain detection using deep learning. pages 139–143.
- Silveira, M. R., Marcos da Silva, L., Cansian, A. M., and Kobayashi, H. K. (2021). Detection of newly registered malicious domains through passive dns. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3360–3369.
- Tanenbaum, A. S., Feamster, N., and Wetherall, D. (2021). *Redes de Computadores*. Pearson Education do Brasil, Porto Alegre, 6 edition.
- The Pandas Development Team (2020). pandas-dev/pandas: Pandas.
- Vranken, H. and Alizadeh, H. (2022). Detection of dga-generated domain names with tf-idf. *Electronics*, 11(3).
- Wang, Y., Pan, R., Wang, Z., and Li, L. (2023). A classification method based on cnn-bilstm for difficult detecting dga domain name. In *2023 IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 17–21.
- Wong, A. D. (2023). Detecting domain-generation algorithm (dga) based fully-qualified domain names (fqdns) with shannon entropy.
- Yu, B., Pan, J., Hu, J., Nascimento, A., and De Cock, M. (2018). Character level based detection of dga domain names. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.
- Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. <https://D2L.ai>.