IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 March 2025, accepted 15 April 2025, date of publication 28 April 2025, date of current version 15 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3565022

RESEARCH ARTICLE

Mixed-Embeddings and Deep Learning Ensemble for DGA Classification With Limited Training Data

CHRISTIAN MORBIDONI[1], **ALESSANDRO CUCCHIARELLI**[2], **AND LUCA SPALAZZI**[2]
[1]Department of Business Administration, Università degli Studi G. d'Annunzio Chieti–Pescara, 65127 Pescara, Italy
[2]Department of Information Engineering, Università Politecnica delle Marche, 60121 Ancona, Italy

Corresponding author: Christian Morbidoni (christian.morbidoni@unich.it)

ABSTRACT Recent papers in the cybersecurity research field of Domain Generation Algorithms (DGAs) detection show the increase of performances associated with the introduction of unsupervised neural vectorized representation of domain names in the supervised classification process. In this paper we explore the effectiveness of this approach by proposing a novel mixed pre-trained neural embeddings model which integrates different vectorized representations of domain names: n-grams streams and words. We used the embeddings with two different classifiers, both based on ensemble architectures: a stacking model and an end-to-end multi-input neural architecture. We trained and tested the classifiers with two datasets, differing both in the distribution of domain names between real and DGAs and in the number and type of DGAs. The obtained results show that our solution provides considerable advantages with respect to state-of-the-art single classifiers both in classification accuracy and in the detection of challenging DGAs, such as those based on word dictionaries. The improvement of performance is significant in a particularly relevant operating condition, known as few-shot-learning, where only few examples of DGA-generated domain names are available for the classifier training.

INDEX TERMS Domain generation algorithms (DGA), botnet, deep learning, LSTM, n-grams, pre-trained embeddings, few-shot learning.

I. INTRODUCTION

Cybercriminal activities aimed at undermining the security of computer systems are increasing in number and intensity year after year [1], [2], [3]. In this context, botnets, i.e. networks of malware-infected systems (bots) operating under the control of a botmaster, are one of the most popular tools used by cybercriminals. A common botnet attack is the Distributed Denial of Service (DDoS). To counter this, it is possible to disrupt botnet operations by identifying and redirecting communications from the botnet's Command & Control (C&C) server [4]. As a countermeasure, bots use Domain Generation Algorithms (DGAs) to produce pseudo-random or word-based domain names shared with the botmaster. Only one of these domains is registered and active for a short time to control the network.

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Sun.

Research on identifying network traffic from DGA-based bots employs two DNS traffic analysis approaches. The first, context-less, examines only domain names to classify them as benign or malicious [5], [6], [7], [8], [9]. The second, context-aware, takes into account features extracted from DNS network traffic like timing, responses, and TTL-based data to detect malicious domains [10], [11], [12], [13]. Despite some challenges of the context-less approach, such as the inability to focus on DNS query patterns of the network traffic produced by botnets, the ability of DGAs to mimic the lexical form of legitimate DNs and the complexity of generation patterns they use, several researches have shown that the context-less approach is more effective because it achieves better global performance and is more efficient in terms of resources used [7], [8], [14].

In the context-less approach, machine learning techniques can be effectively applied to identify DGAs generated DNs in DNS traffic [14], [15], and, in recent years, the effectiveness of deep learning (DL) techniques for classification

has emerged [16], [17], [18]. In particular, LSTM based architectures, as the one proposed in [19], was shown to overcome several machine learning techniques, such as Hidden Markov Model (HMM) [20] and Cost Sensitive Support Vector Machines (CS-SVM) [21].

The price to pay is the need for huge synthetic datasets of labelled data for the training phase of DL methods [22] and these datasets are hard to obtain. Lists of real DNs, used to assess the popularity of sites, are publicly available (see, for example the Majestic Million dataset[1]) and DNs to be used in training datasets can be extracted from them. On the contrary, certainly malicious DNs can only be extracted by reverse engineered DGA[2] [23], [24]. These processes are, however, not scalable [23] and not always feasible in an efficient manner, since the DGAs change very quickly and the bots using them are obfuscated and difficult to detect [20]. An alternative to this method is the collection of malicious DNs from real DNS traffic, considering the ability of DNS servers to resolve a DN or not as a clue to label a DN as legitimate or malicious [23]. But, even using this method, finding large sets of malicious DNs is challenging, both because of the difficulty of discriminating whether an unresolved DN is actually produced by a DGA or is just an incorrect DN, and because malicious DNs are in any case very rare in DNS traffic, as reported in [25].

The problem of training a classifier with limited availability of DNs samples is particularly relevant in the detection of newly appeared DGAs [26] or in case of malware families that generate small numbers of domains per day (e.g. *matsnu* generates only 10 domains daily [27]). This condition, known as few-shot learning (FSL), is of recognized importance in malware detection from network traffic [28], [29], [30], [31].

However, most of the work on FSL so far deals with computer vision [32]. Only very few works have addressed DGA detection. In this context, it should also be noticed that, while recent studies as [31], [33], and [34] suggest that relying on contextual embeddings can increase classification capabilities of a context-less DL classifier for specific DGAs, performances are not uniform across the different types of DGAs. This can be a problem in practical applications as different kinds of DGAs are to be targeted. For example, as emerges from our experiments, while word embeddings work well for dictionary based DGAs, they do not perform optimally with some hard-to-detect char-based DGAs, where 2-grams embeddings are shown to provide better results.

In this paper, we propose the use of context-less mixed-model classifiers implementing an ensemble learning process [35] in order to have good performances with both dictionary-based and non dictionary-based DGAs. The term *ensemble learning* defines methods that combine multiple classifiers to make a decision, typically in supervised machine learning processes. Indeed, we present a new classification model for DGAs generated DNs, represented by unsupervised character-level n-grams (as in [33]) and word embeddings. We trained the character-level n-grams embedding with a dataset of unlabelled DNs.

We experiment with two ensemble models, which implement different fusion strategies: a stacking generalization approach with a multinomial regression fusion layer and an end-to-end multi-channel neural architecture.

Two different datasets are used to evaluate our method in different learning and testing conditions: the 25-DGA dataset[3] defined in [33] and a dataset extracted from UMUDGA [24] with DNs generated by 50 DGAs. They differ in the number of DGA families that generate the DNs and in the ratio between malicious and legit DNS.

With the tests conducted on our classifier models, we answer the following three research questions:

- **RQ1** Do mixed-model classifiers based on contextual pre-trained embeddings perform better than single stage classifiers in the recognition of DGAs generated DNs?
- **RQ2** How does their performance compare with single stage classifiers as the size of the learning set changes?
- **RQ3** Are they more effective in the operational condition of few-shot learning?

Our experiments show that the classifiers have good performances both in the classification accuracy and in the detection of challenging DGAs, such as those based on word dictionaries. They outperform the single stage architecture described in [33] and the method they are based on is, to the best of our knowledge, the only one in current literature on DGAs detection to use both a pre-trained embeddings learned exclusively from unlabeled DNs dataset and an ensemble learning in order to have a multi-class classifier. Moreover, we demonstrate that the proposed mixed-model classifier has significantly better performance than the single-stage architecture, especially for a small learning dataset, and it is therefore also effective in the few-shot learning condition. In this specific context, from the analysis on the most recent studies reported in the next section, emerges that no prior work has explored fusing unsupervised latent representations to evaluate their effectiveness across diverse DGA types in multi-class classification with limited training data.

The rest of the paper is organized as follows. In Section II, we frame our work in the context of the state of the art, reviewing the main related literature. In Section III we illustrate the main steps of our method and present the classification architectures, while in Section IV we describe our experimental settings. Finally, in Section V we discuss the obtained results and draw the final considerations in Section VI.

II. RELATED WORKS

Machine learning, particularly supervised learning, has long proved to be one of the most effective and therefore most

[1] https://it.majestic.com/reports/majestic-million
[2] https://osint.bambenekconsulting.com/feeds/
[3] https://github.com/chrmor/DGA_domains_dataset

widely used approaches to DGA-based malware detection. In this section, the main approaches adopted in the literature and their relative strengths and drawbacks are reviewed.

A. MACHINE LEARNING

In this context, approaches based on shallow learning (i.e. based on features) are often opposed to those based on deep learning (that do not require features engineering).

Features used in literature can be grouped in three categories: statistical features [10], [20], [36], [37], [38], [39], [40], [41], [42], [43], [44], information theory features [36], [37], [40], [44], and lexicographic features [6], [10], [11], [38], [40], [41], [42], [43], [45]. Kostopoulos et al. [44] applied SHAP method [46] in order to establish how features affect the classifier performances.

Conversely, deep learning is often preferred, as hidden features are learnt [22] and has been shown to perform better with large training datasets, the larger the better [22], [38], [47]. Indeed, Li et al. [38] observe that MLP performs better than J48 and LSTM with intermediate-size datasets (from 5000 to 500k samples), whilst LSTM with big size datasets (greater than 500K samples). High performances on multi-class classification can be achieved with distance features informed by the training set, but at a considerable cost in terms of data processing time [6].

Neural architectures applied in literature include Multi Layer Perceptrons (MLP) [38], Convolutional Neural Networks (CNN) [30], [38], [42], [43], [48], [49], [50], [51], Recurrent Neural Networks (RNN), specially Gated Recurrent Units (GRU) [52] and Long-Short Term Memory networks (LSTM) based architectures [19], [47], [48], [49], [50], [51], [53], [54], [55], and, finally, Transformer networks [56], [57], [58]. Recently neural networks have seen the application of attention mechanisms [49], [50], [56], [58], [59], [60] and pre-processing mechanisms with CNN layers [49].

B. EMBEDDING TECHNIQUES

The idea of using pre-trained word embeddings, learned in an unsupervised way, to represent text input was shown to provide advantages in a variety of NLP tasks. In the context of DGA detection, some attempts to leverage pre-trained embeddings to improve classification performances were made in literature [31], [34], [42], [43], [58], [60], [61], [62]. Such techniques can be grouped into word level context aware embedding techniques and n-gram-based embedding techniques.

Word level context aware techniques have been proved to work well with DNs generated by algorithms based on a vocabulary of the same language used by embedding [31], [62]. Yang et al. [63] introduce a pre-processing module to parses DNs and uses different embedding layers for different token types (e.g. words and numbers). A popular context aware word embedding tool turns out to be ELMO [31], [62] trained by Google on the 1 Billion Word Benchmark dataset. However, some authors have proposed embedding algorithms specifically designed or trained for the detection of DGA-based malware. Huang et al. [64] use a one-hot encoding so that a DN is transformed into a 2-D tensor. Similarly, Liang et al. [43] and Chiscop et al. [42] use a Right Shift Tensor (RFT) to feed the classifier with inputs similar to images. Sidi et al. [60] proposed *Helix*, an embedding algorithm based on an autoencoder architecture. This technique allows them to use a large (easy to obtain) unlabelled dataset to train Helix and this allows them to use a small labelled dataset to train the classifier. Huang et al. [34] used *BERT*, a character embedding based on transformer networks. Similarly, Gogoi and Amhed [58] used *CANINE-c*, a pre-trained character embedding also based on transformer networks. Finally, Fang et al. [61] proposed an incremental variant of the well known *Word2Vec*, where a small untagged dataset is used as input to incrementally retrain with new data only. They apply the proposed technique to the network traffic in order to capture temporal patterns of the DNs calls. Of the works mentioned above, only Chiscop et al. [42] and Huang et al. [34] address multi-class classification over a variety of DGA types.

C. FEW-SHOT LEARNING

The need for deep learning to have large labeled datasets clashes with the difficulty encountered in collecting a large number of malicious DN samples, often relying on reverse engineering to trace the domain generation algorithms [14]. Furthermore, such datasets quickly become obsolete, requiring updating and retraining [20], [23] for recently discovered DGA families in a short time and caused the so-called dataset imbalance problem, i.e. the presence of poorly represented classes in the training set [19], [26], [65]. Hence the need to investigate how to deal with the scarcity of malicious samples, the so-called *few-shot learning* (FSL) problem [28], [29], [30], [31]. However, although FSL is becoming more and more relevant [32], most of the works deals with computer vision, only a small part of them deals with natural language processing [66] and even less deals with DGA detection. In fact, although there are some works that emphasize this need for DGA detection, only a few works have proposed solutions for this problem. From these few works it emerges that focusing on the *pre-training* stage at the *feature level*, according to the taxonomy proposed by Song et al. [32], represents the most promising approach. In particular, two types of techniques have been identified: embedding techniques and feature extraction techniques. Embedding techniques for DNs can in turn be classified into two categories: techniques based on n-grams and techniques based on vocabularies.

Regarding the work presented in [31], the authors use a fully-connected layer with 128 rectified linear units followed by a logistic regression output layer. Experimental results are promising even though the classifier has only been trained and tested with 4 families of DGA of the

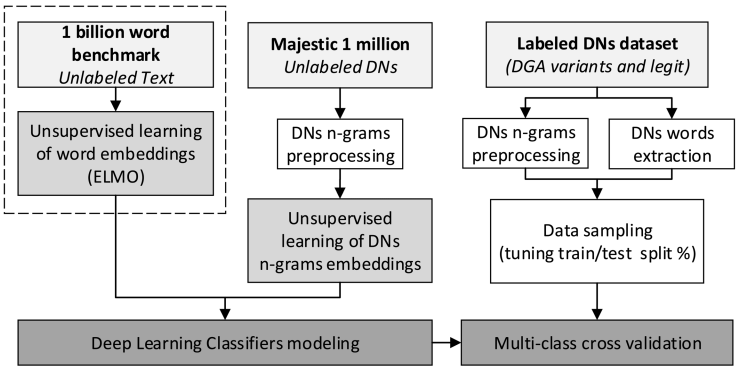

FIGURE 1. Operational workflow.

vocabulary-based type. These experiments show that the use of vocabulary-based pre-trained embedding techniques seem to be appropriate for vocabulary-based DGA families. However, given the limited number of DGA families used, the results neither can be said to be conclusive nor cover multi-class classification with diverse DGAs.

In [33], authors use embedding techniques for n-grams to define which vectors provide as input to a LSTM.MI classifier. They present the experimental results for the cases of 1-grams, 2-grams and 3-grams. The experiments were performed on a dataset with 25 DGA families and the results were compared with those obtained from an LSTM.MI with randomly initialised embedding. The outcomes show that the presence of pre-trained embedding improves the results in the case of character-based DGA families. However, the results for vocabulary-based DGA are lower.

In [34], authors use BERT, a context-depending character-based embedding, to define which vectors provide as input to a Deep Parallel Convolutional Neural Networks (DPCNN) for classification. They trained the classifier on an unbalanced dataset with 20 DGA families (10000 benign domains and 500 malicious domains for each DGA family). They performed their experiments on datasets of increasing size starting from 6% of the original dataset and doubling it up to 96%.

In [67], authors use a Siamese BiLSTM network to build a feature extractor. The feature extractor is then used by a layer of shallow learning binary classifiers, namely a Support Vector Machine, a Random Forest, and a k-Nearest Neighbor. The decision is then taken according to a voting mechanism. The classifier was then trained and tested with a dataset of 2400 legit samples and 2400 malicious samples from 24 DGA families (100 samples for each DGA family).

D. ENSEMBLE CLASSIFICATION

Few recent works combined different models into simple ensemble models.

Both Tran et al. [19] and Tuan et al. [68] applied a simple two-stage model to combine a binary and a multi-class classifier operating on the same input data.

Liang et al. [43] address the problem of sensibility to domain name length. They introduce a conditional classifier that uses different architectures depending on the length of the domain to be classified. Two-class classification experiments on a self-made dataset show promising results.

A conditional (multi-class) classifier is also proposed by Liew and Law [51]. A feature-based classifier or a deep learning classifier is chosen depending on the nature of the domain name, respectively: word-looking or random-looking.

A parallel architecture is proposed by Yang et al. [56] in order to obtain a binary classification. Transformer networks are trained with different n-gram encodings, they work in parallel and their output is combined to obtain the final result.

No attempt exists, to the best of our knowledge, to investigate the fusion of different unsupervised latent representations and to focus on evaluating their effectiveness over a variety of DGA types, in a multi-class classification setting, and with limited training examples.

In summary, the study of these works shows the importance of both studying classification systems based on embedding to tackle the few-shot learning and of combining in an ensemble classifier different techniques such as character-based embedding with vocabulary-based embedding.

III. METHODS

In Figure 1 the general operational workflow adopted in this study is illustrated. In the following subsections we will detail and discuss the different steps depicted and the methods adopted. The workflow is then instantiated in our experimental setting (Section IV).

A. DOMAIN NAMES PRE-PROCESSING

In this work we represent DNs as ordered series of lexical elements. Such elements can be single chars, as in the majority of the previous studies in DGA detection with Deep Learning, or aggregation of chars such as char-level n-grams of fixed length. The basic idea behind n-grams based representation is that some sequences might capture acronyms, abbreviations, short words or, in general, patterns

TABLE 1. Examples of domains processed to derive sequences of n-grams and words.

DGA	Example	
bedep	1-grams	qejnttponl5i
	2-grams	qe ej jn nt tt tp po on nl l5 5i
	3-grams	qej ejn jnt ntt ttp tpo pon onl nl5 l5i
	words	qe j ntt pon l 5 i
cryptolocker	1-grams	jssbrunhmddq
	2-grams	js ss sb br ru un nh hm md dd dq
	3-grams	jss ssb sbr bru run unh nhm hmd mdd ddq
	words	j ssb run hm dd q
fobber	1-grams	tvachuckag
	2-grasms	tv va ac ch hu uc ck ka ag
	3-grams	tva vac ach chu huc uck cka kag
	words	tv a chuck ag
necurs	1-grams	lbyrootcdes
	2-grams	lb by yr ro oo ot tc cd de es
	3-grams	lby byr yro roo oot otc tcd cde des
	words	l by root c des
nymaim	1-grams	renophysical
	2-grams	re en no op ph hy ys si ic ca al
	3-grams	ren eno nop oph phy hys ysi sic ica cal
	words	reno physical
suppobox	1-grams	meatmonday
	2-grams	me ea at tm mo on nd da ay
	3-grams	mea eat atm tmo mon ond nda day
	words	meat monday

employed by a DGA. This, in turn, might help a machine learning model to derive meaningful features. In this setting, DNs are broke down into overlapping series of n-grams, as exemplified for 2-grams and 3-grams in Table 1. In the remaining of this paper we will refer to the char-based representation as 1-grams.

An alternative is that of attempting to find and extract actual words composing the DN. The idea is that capturing common sense words and relying on pre-trained word vectors can be beneficial to characterize DGAs that make use of dictionaries to generate DNs, thus improving detection. While the pre-processing step to produce n-grams sequences is straightforward, extracting words requires a pre-loaded language model and a non trivial tokenization techniques, with inevitable non null error rate. In our experiments, words are extracted using the Wordninja python library,[4] as done in previous studies [31], [62], and the result is exemplified in the last column of Table 1. As one can see, the word based representation leads to possibly erroneous tokens for some DGAs. While in the case of nymaim and suppobox, two well known dictionary based DGAs, the representation clearly captures two of the words used by the algorithm, for the other, char based DGAs, it is hard to say if the extracted tokens (e.g. root, chuck, run) actually capture some meaningful pattern.

B. EMBEDDED VECTORS

As happened with several text classification tasks, also DGA detection has been acknowledged to benefit from embedding vectors representation of DN elements. Such embeddings are usually pre-trained in an unsupervised manner from a large corpus of unlabeled text documents. In the case of word-based representation, pre-trained ELMO word embeddings have been evaluated in a binary classification task where dictionary-based DGAs only were considered [31], [62]. However, in a multi-class classification problem, we might want to consider a variety of DGA types. We will see later that relying on words it is not the optimal solution for non dictionary-based DGAs. In [33] the idea of relying on an unlabeled DN dataset for learning embeddings to represent n-grams was introduced. Preliminary results show that 2-grams and 3-grams do in fact provide advantages in multi-class classification with small training sets. However, neither this study considers words nor attempts to combine the different representations and corresponding embedded vectors, as done in the present paper.

In this paper we consider to two types of embeddings:

- **n-grams embeddings.** We ran a skip-gram model over the Majestic Million dataset, which includes one million of the most requested legitimate DNs to derive 128 elements vectors representing 1-grams, 2-grams and 3-grams.To do this we relied on the FastText implementation. The skip-gram model is commonly used over a corpus of text documents, each one composed by space-separated words. In order to adapt our dataset to this model, we pre-processed the DNs to obtain, for each DN, a text document where words are the consecutive n-grams composing the domain, separated by a space, as exemplified in Table 1. During the skip-gram training we used a softmax activation and a cross-entropy loss, we set the learning rate the default value used by Fasttext (0.1) and ran the training for 20 epochs. The context window length was set to 5.
- **word embeddings.** Consistently with recent literature we based on the ELMO pre-trained word embeddings available from Tensorflow Hub,[5] providing word embeddings of 1024 elements trained on the *1 Billion Word Benchmark* [69] as described in [70].

C. CLASSIFIERS ARCHITECTURES

Different Artificial Neural Network (ANN) architectures have been proposed to address the DGA classification problem; a common component of such architectures is a LSTM network, which is capable to learn features from a series of values. Therefore, as in our study we are interested in evaluating the contribution of pre-trained vectors with different amounts of labeled domains, we decided to adopt a common architecture type, where features extracted by an LSTM are processed by a fully connected layer to predict one of the classes corresponding to different DGA variants, plus a ''legit'' class that contains legitimate domain names.

For each domain representation, we implemented an ANN, taking as input a series of lexical elements, composed by an embedding layer providing input vectors to a LSTM network

[4]https://github.com/keredson/wordninja

[5]https://tfhub.dev/google/elmo/3

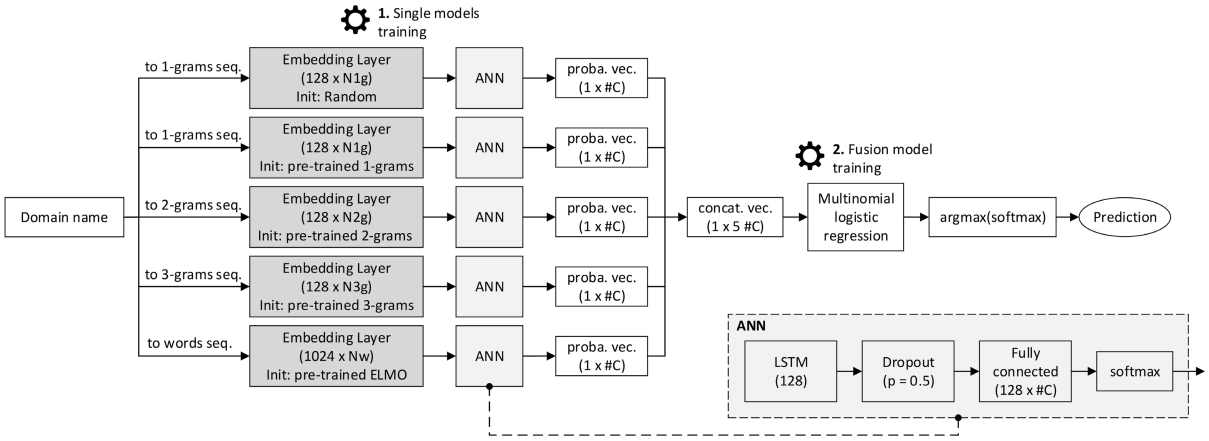

FIGURE 2. Stacked model.

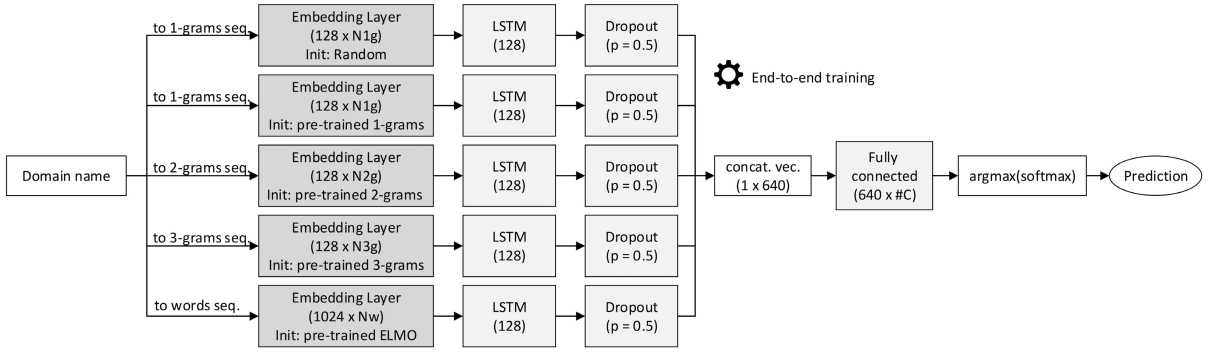

FIGURE 3. Multi-input model.

with an hidden state of 128 units, a dropout layer (with dropout probability set to 0.5) to foster regularization, and a fully connected layer followed by a softmax activation. This simple architecture is also adopted by authors of [19] and proved to provide state of the art results among deep learning approaches on both 25-DGA and UMUDGA [6]. We refer to such models as single models. We thus consider 4 single models, each one initialized with different pre-trained token embeddings (1-grams, 2-grams, 3-grams and words). The single models architectures are identical, except for the embedding layer dimensions, which depends on the tokens type. In addition, we also consider a randomly initialized model that is identical to the 1-grams model but with random embeddings initialization. This is equivalent to the model used in LSTM.MI [19], that we use as baseline model (the Random init. model in tables and figures). We then considered two different architectures to fuse the 5 single models into a mixed model, capable of relying on different domain representations and corresponding pre-trained embeddings.

In the first architecture, we applied a classical machine learning stacking ensemble technique [71]. In the stacked model, there are two distinct stages: in the first stage each one of the 5 previously discussed single models is trained. In the second stage the predictions from the trained models from stage one are used to train a classifier (called meta model), which provides the final predictions. The meta model is composed by a multinomial regression that receives the concatenation of the prediction vectors from all the single models, as depicted in Figure 2.

In the second model (Multi-input), the integration among different vectorized representations is operated at hidden features level, thus having a single multi-layered LSTM-based neural network. Features based on n-grams and word embeddings are learned by parallel LSTM networks and then concatenated to be fed to a Fully connected layer to learn weights for the classification task. The detailed architecture is illustrated in Figure 3.

IV. EXPERIMENTAL SETTINGS

In this section, we detail the design of our experiments, discussing the datasets used for evaluation, the training of the models and the evaluation metrics adopted.

A. DATASETS

In our experiments, we based on two datasets used in previous studies: the 25-DGA [6] and the UMUDGA [24] datasets.

TABLE 2. Overview of the subsets of the 25-DGA dataset and of the UMUDGA dataset used in our experiments.

25-DGAs Train subsets
classes: 1 legit, 25 DGAs
legit DNs: 50% of the total

	Overall	# DNs per DGA
T-17	844	∼17
T-34	1687	∼34
T-68	3360	∼68
T-135	6720	∼135
T-270	13480	∼270
T-540	27000	∼540

UMUDGA Train subsets
classes: 1 legit, 50 DGAs
legit DNs: 2% of the total

	Overall	# DNs per DGA
T-17	867	∼17
T-34	1734	∼34
T-68	3468	∼68
T-135	6885	∼135
T-270	13770	∼270
T-540	27540	∼540

TABLE 3. Overview of the classes included in the 25-DGA dataset with randomly chosen DN examples.

Char-based DGAs	
conficker	akdtfkh.com.bs, flqxktto.co.za, hiroulyv.mu
corebot	a2ix16edy61f1xg.ddns.net, 1ruvwjy4u850385.ddns.net
cryptolocker	cryptolocker,ymirohscgjxm.ru, ugjywfellfju.co.uk
emotet	vtypqtkjnkyfycwj.eu, byoljbjregchfbtd.eu
murofet	nirauhusfeormfuwdovrrfinj.biz, jxqtupnmsmknn.info
necurs	wjilrilcim.ru, gfqcpaifkxd.bit
padcrypt	adcaeccnacfnccma.com, bnmfladdaccalkff.com
qadars	8igi8qwu468u.com, e7wx2jkl2zw5.net
symmi	qaliesvomo.ddns.net, meakugulu.ddns.net
dircrypt	zqmyueidka.com, rafspijepwfkwna.com
fobber	guzhbhepwd.com, dxpgomjalm.com
kraken	gqnysr.dyndns.org, qqvgpzewpaj.com
pushdo	suxqeaki.ru, hilgeaki.ru
pykspa	cgyooueoya.biz, hhqaiifox.com
ramdo	ciqauuaieywkkams.org, eiecsgkuamkwscyg.org
ramnit	aimbvmwceer.com, temhosjjrcini.com
ranbyus	aesxwvhepgpsglydp.su, ibcavldgocpkyegbq.cc
rovnix	46gfypffuuasknujay.ru, vscfnykbbbe4h2wx5y.com
simda	hacudaz.su, wycinep.net
tinba	impxyrifhmld.biz, hwwjgnhhryry.com
vawtrak	fusirnat.com, maduhgumde.com
Dictionary-based DGAs	
gozi	animargumenta.com, surfacepapanobi.com
matsnu	branch-tower.com, film-water-image.com
nymaim	smilefavorite.uz, soft-professor.com
suppobox	groupfirst.net, brooklynnwashington.net
Legitimate domains	
alexa	perrosguru.ru, nhst.no, americanphotomag.com

The 25-DGA dataset includes 25 different DGA families and is balanced among legitimate and DGA generated domains, meaning that there is a number of legitimate domains (taken from Alexa, a list of the most popular web sites provided by Amazon and no longer available since May 2022) equal to the sum of domains over all the DGAs. The number of domains is also balanced across DGA families. Being the DGA families 25, the ratio between the cardinality of the legit family and the cardinality of each DGA family is 25:1.

In Table 3 we show some randomly extracted examples of DNs captured by the 25-DGA dataset, including 4 dictionary based DGAs and a variety of different char-based families.

The UMUDGA is a recent benchmark dataset introduced in [24], and includes 50 different DGA variants belonging to 37 distinct families.

The dataset is balanced across all the classes, variants and legitimate domains class. In other words the number of legitimate domain is equal to the number of domains produced by each one of the distinct DGA variants. In addition, the set of legit domains also contains examples of challenging DNs, as randomly generated DNs (i62e2b4mfy.com, jf71qh5v14.com, ymzrrizntbhde.com,, ...) and short DNs (mag2.com, adyen.com, gds.it, ren.tv, ...). The authors published different dataset *tiers*, including a different number of examples per class. For the purpose of our experiments, we consider the *1k tier*, which includes 1000 DNs for each of the 50 variants.

As the two datasets differ both in number or classes and in the proportion of training data between legit and DGA, they can be used to challenge a DGA classification system differently. On the one hand, as 50 different DGAs are captured in UMUDGA, this makes multi-classification more challenging. On the other hand, since legit represents the largest class in the 25-DGA dataset, this should facilitate the detection of legitimate DN, as indeed later confirmed by our experiments.

In both cases we partitioned the data in order to evaluate the models with different train sets of different sizes. For the 25-DGA datasets, to remain consistent with a recent study [33], we based on the same data sampling mentioned in the paper. In particular we used the 4 smallest sub-sets referred to in [33]. In addition, in order to investigate a few-shot condition, we added two further datasets, smaller than the previous ones, respectively with 34 and 17 training examples for each DGA.

We proceeded in a similar way with UMUDGA. Starting from the 1K tier version of the dataset, we varied the proportion between training and test set in order to evaluate the models in similar conditions as those adopted in the 25-DGA experiments (i.e. number of training examples per DGA).

An overview of subsets is provided in Table 2. According to the number of different DNs included in the train set for each DGA (17, 34, 68, 135, 270 and 540), we label the subset T17, T-34, T-68, T-135, T-270 and T-540, respectively. In each single experiment we trained the classifier with a small portion of the entire datasets leaving unvaried the proportion among different classes, and then tested the trained classifier on the remaining domains.

B. TRAINING THE CLASSIFIERS

For each subset of 25-DGA and UMUDGA, and for each classifier, single models and mixed models, a K-fold evaluation with stratified sampling was performed. In single models we adopted the LSTM architecture evaluated in [19]

as it provides reliable results in dataset where some DGAs are poorly represented. To avoid bias in evaluating the contribution of pre-trained embeddings, we used the same hyper parameters used in [19].

A total of 20 folds were run for each dataset, obtaining a standard deviation of F1 score across the folds ranging from of 0.01 in T-17 to 0.007 in T-540.

While all single models and the Multi-input model can be trained in an end-to-end fashion, the training procedure for the Stacking model requires different steps. Specifically, for each one of the 20 folds, the following steps were performed:

- The training set was split into two disjoint sets: set A (80%) and set B (%20);
- Each single model was trained on set A;
- Each trained single model was tested on set B, producing prediction probability vectors (as output of the softmax function);
- The steps above were repeated 20 times, each time choosing different, non overlapping set B, in a 20-fold validation setting. In this way, at the end of the 20 iterations we obtained, for each single model, the predictions for the entire training set.
- The predictions of the single models obtained in the previous steps were then concatenated and used as feature vectors to train the meta model (a logistic regressor) for the target class prediction;
- Each single model is than trained again on the entire training set, in order to leverage all the available data examples;
- Finally, the whole ensemble is evaluated against the test set. Specifically, each test domain is provided as input to each single model producing 5 probability vectors; they are then concatenated and provided as input to the logistic regression predictor, which outputs the final prediction.

The same hyper-parameter set was used to train both the single LSTM-based models, that compose the stacked model, and the Multi-input model. Specifically, the number of epochs was set to 20, the batch size was set to 128, and the RMSProp algorithm with a sparse cross entropy loss function was used for optimization, setting the learning rate to 0.001 and the discount factor to 0.9.

C. EVALUATION METRICS

To evaluate the overall performances in capturing diverse DGAs we mainly rely on accuracy, but we also measure macro and weighted average F1-score.

To evaluate the performances at single class level we measure precision, recall and F1-score for each class, even if, for brevity, we only report F1-score in result tables.

Apart from accuracy and F1-score, a DGA detection system is often evaluated in terms of False Positive Rate (FPR) and False Negative Rate (FNR). In literature, this is done usually in the two-class classification task, where DGAs are labeled as 1 and legitimate domain as 0. In this context, FPR measures the fraction of legitimate domains that are detected as DGA, thus potentially disturbing the normal behaviour of users and legitimate applications once a detection system is put in place. The same metrics can also be defined in the case of multi-class classification, labelling a DN as malicious if associated with a DGA class.

Some papers [30], [72], [73] suggest that a low FPR is very important in deployed DGA detection systems, because blocking legitimate traffic is highly undesirable, while others relate FPR and FNR desired values to the practical application scenarios. In [74], authors distinguish between high-risk scenarios, e.g. where a complete domain-flux attack already took place, and less risky situations, e.g. where the bots are still trying to connect to a C&C server. In the first case a higher FPR might be tolerated, as the main intent is that of blocking the bots. In the second situation, on the contrary a higher FNR might be tolerated, as the priority is to consent normal functioning. In this case a low FPR is desirable [7], [48]

In order to analyse the most challenging DGAs, i.e. those having relatively high chance of being confused with legitimate domains, in the result evaluation section, we breakdown the FNR over the different DGAs, by calculating, for each DGA, the fraction of domains generated by such a DGA that are classified as legit. In a similar way we breakdown the FPR by calculating the fraction of legitimate domains that are predicted as being generated by such specific DGA.

V. RESULTS AND DISCUSSION

In this section we present and discuss the results obtained on the two evaluation sets 25-DGA and UMUDGA.

In general, we observed a consistent behaviour across the two different experimental settings, showing how mixing embeddings provides in general better performances both in few-shots learning and in larger datasets, and across a large variety of DGAs types.

A. 25-DGA DATASET

In Figure 4 we illustrate the results obtained on the multi-class classification task over the different subsets of the 25-DGA dataset.

Specifically, in Figure 4 a) the overall classification accuracy over the different training sizes is illustrated. While all the classifiers, unsurprisingly, show a visible improvement as more data is available for learning, the performances of 1-grams based single models, both random and pre-trained, are poor in small datasets. Among the single models, more stable performances are provided by the 2-grams pre-trained single model, even if such an advantage over 1-grams models vanishes in larger datasets. Looking at the word-based model (in red), one can see that, while providing some gain over the 1-grams models in T-17 and T-34, it has the lowest accuracy in the largest datasets.

The two mixed models provide best results in correspondence of all the training set sizes, providing improvements over single models ranging from 0.16, if compared to 1-grams

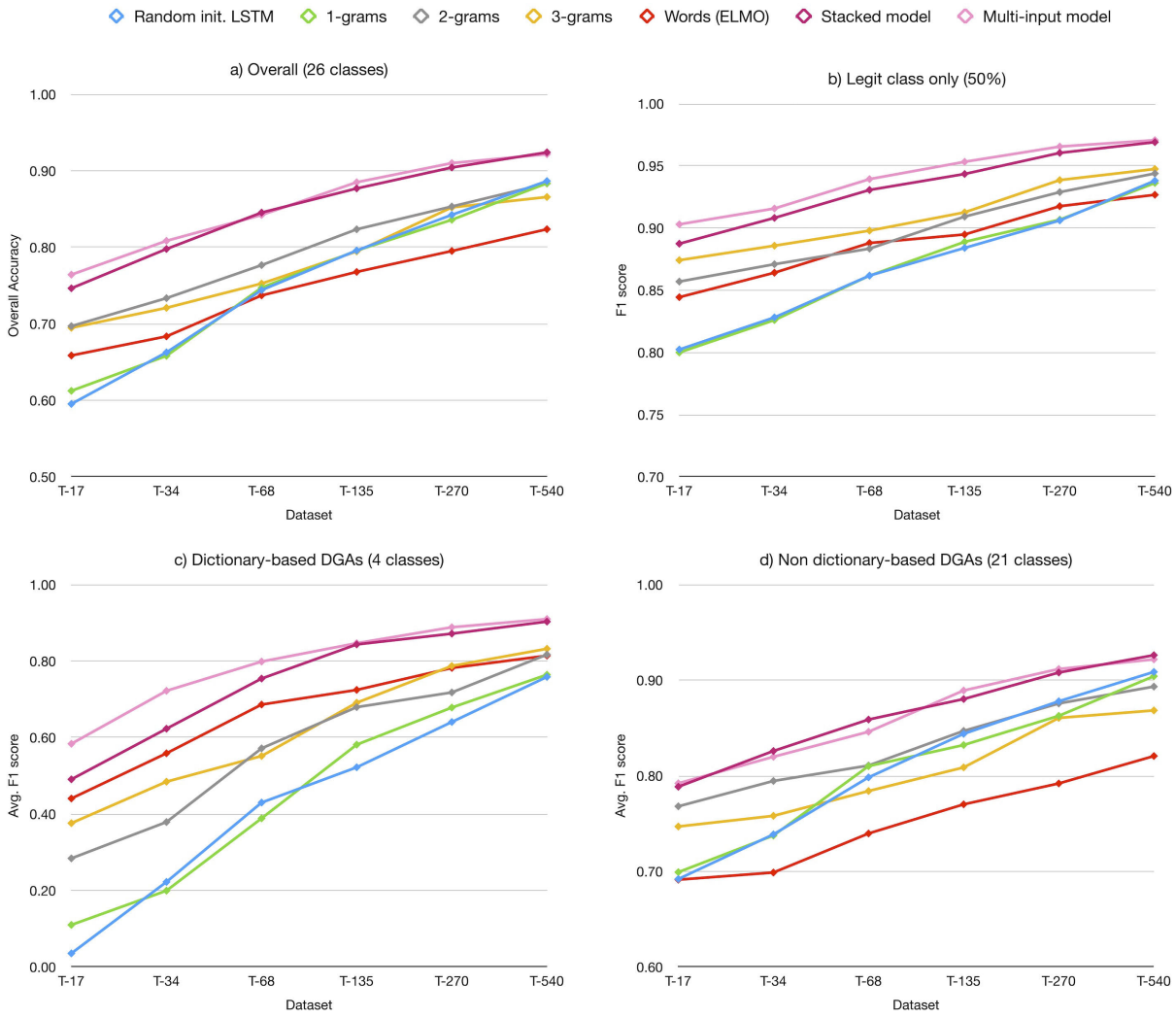

FIGURE 4. Overview of the results obtained on multi-class classification on the different subsets of 25-DGA dataset. For all compared methods and for each differently sized tier, we show - from top-left to bottom-right: the overall accuracy, the F1 score for the legitimate domains (Legit class), the avg. F1 score for dictionary-based DGAs and, finally, the avg. F1 score for the remaining DGAs.

models in T-17, to 0.03 in the case of the largest dataset investigated (T-540). This first results seems to confirm the fusion capacity of the ensemble models. While results register a small preference for the Multi-input model, as its overall accuracy is slightly higher that the stacked model's one in small datasets, we can observe very similar performances between the two.

In Figure 4 b), c) and d) we illustrate the results focusing on specific DGA classes. They are, respectively, the legit class in b), the 4 dictionary based DGAs in c) and the remaining char-based DGAs in d). As expected, the word based model over-performs the other single models in dictionary-based DGAs. Results on the non dictionary-based DGAs in d), reveal that the word based model significantly loses discriminative power on non-dictionary based algorithms. The reason could be that the advantage provided by the pre-trained word embeddings, that encode context-based similarity among words, provides a notable advantage when few domains are seen during training, while such an advantage vanishes when the network is able to learn meaningful features from more data. In all the analysed cases the best performances over all the tiers are still produced by the mixed models. Furthermore, results suggest that the Multi-input model in more effective to detect dictionary-based DGA in few-shots conditions.

1) DETAILED RESULTS

Per-class results obtained on the T-17 and T-540 25-DGA subset are shown in Figure 5, where the cells are colored from dark orange (lower values) to dark green (higher values).

In the few-shot condition, T-17 (Figure 5 (a)), a considerable amount of DGAs fail to be correctly classified, especially hard-to-detect DGAs like *necurs*, *ramnit*, *dircrypt* and *cryptolocker*. However, the 2-grams and 3-grams models succeed in detecting 6 classes with a F1 score over 0.85. The results get sensibly better in T-540 (Figure 5 (b)), but poor results still remain (e.g.*dircrypt*, *ramnit* and *cryptolocker*).

	Single models					Ensembles	
	Random init.	Pre-tran. 1-grams	Pre-train. 2-grams	Pre-train. 3-grams	Words (ELMO)	Stacked	Multi-input
corebot	0.660	0.810	0.950	0.950	0.890	0.990	0.990
symmi	0.550	0.690	0.900	0.940	0.890	0.980	0.980
qadars	0.480	0.580	0.920	0.940	0.750	0.950	0.960
padcrypt	0.150	0.450	0.900	0.910	0.440	0.960	0.950
ramdo	0.450	0.580	0.890	0.890	0.690	0.930	0.950
alexa	0.800	0.800	0.860	0.870	0.840	0.890	0.900
rovnix	0.370	0.590	0.780	0.560	0.670	0.890	0.870
matsnu	0.120	0.410	0.480	0.530	0.730	0.850	0.840
pushdo	0.040	0.020	0.670	0.720	0.290	0.750	0.780
emotet	0.570	0.360	0.680	0.300	0.410	0.620	0.710
simda	0.070	0.050	0.510	0.600	0.400	0.580	0.640
pykspa	0.000	0.010	0.410	0.540	0.250	0.540	0.630
vawtrak	0.000	0.000	0.280	0.570	0.150	0.540	0.620
murofet	0.310	0.480	0.560	0.520	0.360	0.630	0.610
suppobox	0.020	0.020	0.430	0.390	0.580	0.590	0.610
gozi	0.000	0.010	0.170	0.470	0.270	0.510	0.600
conficker	0.060	0.080	0.270	0.240	0.260	0.240	0.450
kraken	0.010	0.020	0.320	0.350	0.130	0.360	0.380
ranbyus	0.130	0.180	0.250	0.190	0.220	0.280	0.360
fobber	0.220	0.250	0.310	0.170	0.200	0.320	0.350
nymaim	0.000	0.000	0.060	0.110	0.180	0.010	0.280
tinba	0.130	0.140	0.240	0.190	0.100	0.170	0.260
cryptolocker	0.060	0.090	0.190	0.130	0.190	0.170	0.250
dircrypt	0.210	0.150	0.250	0.160	0.180	0.300	0.250
ramnit	0.110	0.110	0.190	0.150	0.180	0.140	0.220
necurs	0.040	0.040	0.150	0.140	0.120	0.080	0.210
accuracy	0.595	0.612	0.697	0.695	0.659	0.746	0.764
macro avg.	0.214	0.266	0.485	0.481	0.400	0.549	0.602
weighted avg.	0.496	0.522	0.663	0.670	0.613	0.711	0.746

a) 25-DGA T-17

	Single models					Ensembles	
	Random init.	Pre-tran. 1-grams	Pre-train. 2-grams	Pre-train. 3-grams	Words (ELMO)	Stacked	Multi-input
emotet	0.992	0.988	0.988	0.975	0.815	0.994	0.997
corebot	0.996	0.991	0.996	0.997	0.989	0.997	0.996
symmi	0.994	0.994	0.993	0.996	0.991	0.996	0.996
ramdo	0.994	0.992	0.987	0.980	0.954	0.996	0.995
padcrypt	0.990	0.992	0.995	0.983	0.915	0.998	0.993
qadars	0.984	0.977	0.979	0.961	0.956	0.991	0.993
rovnix	0.977	0.978	0.953	0.863	0.941	0.984	0.987
pushdo	0.935	0.936	0.946	0.932	0.772	0.963	0.979
matsnu	0.868	0.868	0.897	0.896	0.946	0.967	0.972
alexa	0.938	0.936	0.944	0.947	0.927	0.969	0.971
suppobox	0.788	0.808	0.869	0.893	0.902	0.944	0.949
simda	0.931	0.954	0.939	0.928	0.785	0.975	0.946
pykspa	0.923	0.921	0.942	0.929	0.801	0.966	0.945
vawtrak	0.948	0.939	0.906	0.927	0.667	0.963	0.943
gozi	0.739	0.700	0.798	0.858	0.774	0.899	0.917
ranbyus	0.851	0.839	0.817	0.686	0.556	0.881	0.892
murofet	0.837	0.834	0.796	0.732	0.716	0.849	0.854
kraken	0.799	0.804	0.808	0.736	0.676	0.846	0.839
conficker	0.723	0.727	0.765	0.671	0.617	0.786	0.803
nymaim	0.639	0.680	0.704	0.681	0.634	0.802	0.802
necurs	0.681	0.681	0.723	0.515	0.320	0.785	0.798
tinba	0.727	0.707	0.706	0.620	0.320	0.794	0.795
fobber	0.815	0.786	0.650	0.552	0.527	0.848	0.779
cryptolocker	0.685	0.663	0.641	0.604	0.498	0.735	0.721
dircrypt	0.458	0.478	0.436	0.381	0.321	0.504	0.496
ramnit	0.431	0.454	0.404	0.308	0.299	0.478	0.465
accuracy	0.887	0.884	0.883	0.866	0.824	0.924	0.922
macro avg.	0.832	0.832	0.830	0.790	0.716	0.881	0.878
weighted avg.	0.883	0.882	0.885	0.866	0.817	0.923	0.922

b) 25-DGA T-540

FIGURE 5. Results of the multi-class classification over the 25-DGA T-17 (a) and T-540 (b). We report F1 scores for each DGA class obtained with the single models and with the two ensemble models, Stacked and Multi-input. We also report, in the last three rows, the accuracy and the macro and weighted average F1 score.

The 2-grams model provides relatively good results in general, (F1 $\geq$ 0.80) on 18 out of 26 classes, while other models provide better results for some specific DGA classes. For example the randomly initialized LSTM model outperforms all the others on *fobber*, *murofet* and *ranbyus*, while the 3-grams pre-trained model is the best at recognising *gozi*. The ELMO based model provides the higher F1 score for *matsnu* and *suppobox*. Such non-uniform results obtained with the single models did in fact motivate the idea of combining them into a single classifier to improve results: the right part of Figure 5 (a) and (b) (Ensembles) confirms this intuition as both the Stacked and the Multi-input models provide higher F1 score for all the classes, thus succeeding in getting the best from each single model.

Considering dictionary based DGAs, the performances obtained with word embeddings on *matsnu* and *suppobox* are sensibly better, while the remaining two dictionary based DGAs (*nymaim* and *gozi*) are harder to detect basing on words. In the case of *gozi* the 3-grams based model performs slightly better. This can be possibly attributed to the specific Latin dictionary used in the DNs generation algorithm (we will later encounter different variants of *gozi* in the UMUDGA dataset). As the ELMO embeddings were trained on English texts, its ability to capture meaningful tokens in this situation is reduced. On the contrary, results suggests that 3-grams can capture some meaningful char sequences within words in a language-independent manner.

B. UMUDGA DATASET

The overall classification accuracy obtained with UMUDGA using a growing number of training examples are summarized in Figure 6 a). In line with the results on 25-DGA, mixed methods beat single models for all sizes of the training set, especially for the smallest ones, where the gain with respect to the best single model (2-grams) is more that 0.1. We measured, however, sensibly lower values, due to the larger number of classes (51 vs. 26 of the 25-DGA dataset) and the presence of few legitimate domains (2%), which makes the task more challenging with respect to 25-DGA.

Considering single models, we observed a trend consistent with the results on the 25-DGA, with 2-grams succeeding in capturing a variety of DGA. Looking at the chart we notice that from T-135 on the 1-grams model perform better than both 3-grams and word based ones. This is probably related to the fact that, with more training samples the fine granularity of the representation becomes more relevant than the contextual similarity captured by longer tokens (especially words).

Looking at Figure 6 b) (Legit domains), we notice lower F1 values if compared to the 25-DGA dataset.

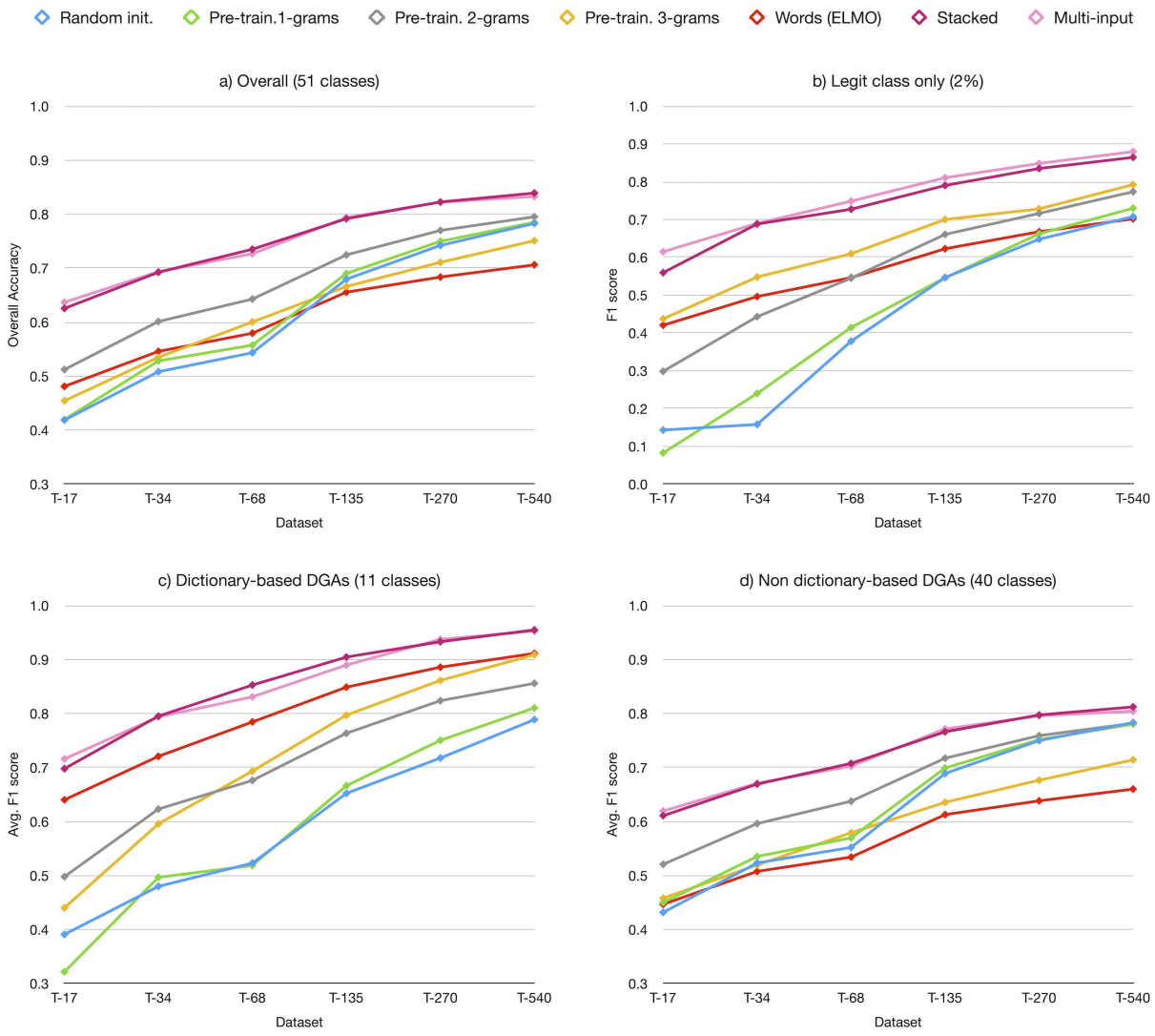

FIGURE 6. Overview of the results obtained on multi-class classification on the different subsets of UMUDGA. For all compared methods and for each differently sized tier, we show - from top-left to bottom-right: the overall accuracy, the F1 score for the legitimate domains (Legit class), the avg. F1 score for dictionary-based DGAs and, finally, the avg. F1 score for the remaining DGAs.

This is also due to the fact that the legit class in the UMUDGA dataset has the same number of examples associated to each DGA family (2% of the dataset size). As happens with the 25-DGA dataset, the Multi-input model seems to work sligthly better on Legit domains detection if compared to the Stacked model.

Among the single models, the 3-grams-based one is the best at detecting legitimate domains, as observed in the 25-DGA, but substantially equals the performances of the 2-grams model with more training examples (T-540). On the contrary, the mixed models retain an advantage over the single components.

The word embeddings model outperforms the other single models on dictionary-based DGAs, especially on small datasets (Figure 6 c) and d)). On the contrary, it provides the worst results on non dictionary-based DGAs, with the exception of the smaller dataset.

1) DETAILED RESULTS

In Figure 7, detailed F1-scores obtained with UMUDGA T-17 and T-540 datasets are shown, coloured in a range between green (higher F1) and orange (lower F1). In general, we can see that the ensemble methods improve detection capabilities over a wide range of DGAs both with few training samples and with more training data.

In the T-17 dataset, the gain in accuracy and macro avg. F1 with respect to the random initialization baseline are 0.22 and 0.25 respectively. These result confirms the effectiveness of fusing different neural DNs vectorizations, but also indicates that there are several DGAs very hard to recognize with limited learning examples. While the situation improves in T-540, there still are few DGAs variants that evade detection. Among them are *ramnit*, *pykspa*, *pykspa_noise*, *dicrypt*. Two of them (*dicrypt* and *ramnit*) confirm the difficulty of detection already shown in the 25-DGA experiment.

	Single models					Ensembles	
	Random init.	Pre-tran. 1-grams	Pre-train. 2-grams	Pre-train. 3-grams	Words (ELMO)	Stacked	Multi-input
murofet_v3	0.869	0.927	0.941	0.916	0.890	0.992	0.997
banjori	0.757	0.787	0.969	0.994	0.966	1.000	0.995
dyre	0.746	0.937	0.942	0.797	0.948	0.994	0.991
corebot	0.809	0.839	0.931	0.911	0.843	0.984	0.987
symmi	0.556	0.599	0.920	0.930	0.855	0.983	0.985
suppobox_3	0.690	0.509	0.769	0.724	0.855	0.947	0.984
vawtrak_v1	0.671	0.590	0.942	0.831	0.832	0.969	0.970
ccleaner	0.867	0.883	0.940	0.814	0.946	0.991	0.965
zeus-newgoz	0.778	0.757	0.763	0.524	0.494	0.972	0.964
padcrypt	0.650	0.815	0.894	0.910	0.435	0.971	0.957
sisron	0.297	0.422	0.899	0.811	0.751	0.944	0.944
vawtrak_v2	0.306	0.247	0.522	0.897	0.355	0.942	0.936
ramdo	0.666	0.700	0.865	0.746	0.613	0.907	0.920
vawtrak_v3	0.228	0.307	0.445	0.813	0.346	0.833	0.885
gozi_gpl	0.521	0.334	0.683	0.425	0.761	0.871	0.868
murofet_v1	0.796	0.799	0.717	0.505	0.545	0.826	0.850
pushdo	0.335	0.408	0.662	0.754	0.439	0.806	0.836
matsnu	0.478	0.475	0.678	0.557	0.788	0.872	0.831
murofet_v2	0.637	0.645	0.681	0.368	0.673	0.734	0.814
nymaim	0.324	0.222	0.506	0.365	0.660	0.798	0.807
rovnix	0.491	0.455	0.390	0.348	0.751	0.788	0.796
gozi_luther	0.294	0.124	0.342	0.511	0.591	0.711	0.787
suppobox_2	0.509	0.481	0.639	0.588	0.568	0.748	0.784
chinad	0.618	0.558	0.525	0.229	0.410	0.775	0.743
qadars	0.526	0.436	0.666	0.543	0.403	0.719	0.725
simda	0.346	0.380	0.621	0.663	0.514	0.652	0.725
kraken_v1	0.165	0.223	0.573	0.577	0.300	0.676	0.687
fobber_v1	0.635	0.627	0.468	0.272	0.435	0.610	0.673
suppobox_1	0.346	0.275	0.464	0.450	0.498	0.551	0.586
gozi_nasa	0.100	0.214	0.302	0.305	0.570	0.455	0.552
shiotob	0.188	0.211	0.382	0.202	0.401	0.522	0.538
pizd	0.317	0.259	0.375	0.374	0.500	0.433	0.465
gozi_rfc4343	0.232	0.192	0.328	0.196	0.500	0.502	0.416
bedep	0.330	0.335	0.298	0.143	0.289	0.383	0.404
proslikefan	0.283	0.259	0.255	0.184	0.192	0.337	0.396
ranbyus_v2	0.179	0.141	0.204	0.121	0.145	0.315	0.369
kraken_v2	0.052	0.080	0.239	0.183	0.199	0.142	0.314
tempedreve	0.122	0.125	0.275	0.149	0.091	0.166	0.291
ranbyus_v1	0.236	0.143	0.097	0.101	0.079	0.190	0.267
pykspa_noise	0.084	0.094	0.122	0.126	0.139	0.144	0.232
alureon	0.299	0.185	0.226	0.169	0.189	0.296	0.220
cryptolocker	0.094	0.150	0.105	0.053	0.093	0.070	0.213
tinba	0.050	0.036	0.149	0.135	0.026	0.046	0.204
qakbot	0.044	0.018	0.148	0.113	0.059	0.055	0.198
fobber_v2	0.091	0.273	0.231	0.145	0.085	0.266	0.195
ramnit	0.046	0.073	0.134	0.093	0.130	0.133	0.190
necurs	0.010	0.031	0.113	0.069	0.090	0.082	0.188
pykspa	0.061	0.069	0.168	0.143	0.141	0.178	0.175
dircrypt	0.118	0.148	0.127	0.114	0.124	0.151	0.168
locky	0.064	0.035	0.173	0.117	0.105	0.161	0.139
accuracy	0.419	0.420	0.512	0.454	0.481	0.625	0.637
macro avg.	0.374	0.371	0.492	0.440	0.452	0.591	0.622
weighted avg.	0.374	0.371	0.492	0.440	0.452	0.591	0.622

a) UMUDGA T-17

	Single models					Ensembles	
	Random init.	Pre-tran. 1-grams	Pre-train. 2-grams	Pre-train. 3-grams	Words (ELMO)	Stacked	Multi-input
banjori	1.000	1.000	1.000	1.000	0.999	1.000	1.000
dyre	1.000	1.000	0.998	0.991	0.996	1.000	1.000
zeus-newgoz	0.999	0.998	0.996	0.962	0.963	1.000	1.000
ccleaner	0.999	0.999	0.998	0.991	0.995	1.000	0.999
murofet_v3	0.999	0.998	0.997	0.996	0.996	0.999	0.999
ramdo	0.992	0.999	0.991	0.966	0.945	1.000	0.999
sisron	0.997	0.998	0.999	1.000	0.989	1.000	0.999
corebot	0.996	0.995	0.997	0.994	0.990	0.998	0.998
symmi	0.994	0.994	0.997	0.994	0.994	0.998	0.998
vawtrak_v1	0.995	0.995	0.995	0.993	0.983	0.998	0.998
suppobox_3	0.961	0.965	0.981	0.994	0.986	0.998	0.997
vawtrak_v2	0.892	0.972	0.991	0.988	0.662	0.998	0.995
padcrypt	0.988	0.992	0.981	0.981	0.932	0.998	0.992
suppobox_2	0.913	0.930	0.959	0.981	0.992	0.997	0.992
murofet_v1	0.988	0.984	0.987	0.977	0.923	0.993	0.991
vawtrak_v3	0.907	0.967	0.963	0.985	0.725	0.995	0.989
pushdo	0.920	0.938	0.960	0.955	0.841	0.991	0.988
qadars	0.941	0.937	0.963	0.895	0.753	0.974	0.985
chinad	0.977	0.967	0.928	0.776	0.820	0.976	0.982
gozi_gpl	0.894	0.896	0.912	0.935	0.947	0.973	0.976
simda	0.952	0.957	0.955	0.933	0.811	0.984	0.972
matsnu	0.913	0.906	0.909	0.916	0.915	0.965	0.968
gozi_luther	0.739	0.748	0.843	0.909	0.870	0.951	0.964
suppobox_1	0.842	0.909	0.910	0.978	0.952	0.983	0.962
pizd	0.804	0.903	0.867	0.952	0.934	0.979	0.956
rovnix	0.754	0.753	0.805	0.861	0.938	0.948	0.954
nymaim	0.874	0.879	0.887	0.871	0.877	0.945	0.939
shiotob	0.886	0.877	0.847	0.596	0.667	0.912	0.924
murofet_v2	0.914	0.906	0.894	0.828	0.858	0.922	0.917
gozi_nasa	0.438	0.503	0.700	0.818	0.823	0.891	0.893
gozi_rfc4343	0.543	0.522	0.642	0.783	0.790	0.873	0.877
fobber_v1	0.893	0.876	0.866	0.745	0.712	0.895	0.857
ranbyus_v1	0.800	0.743	0.762	0.565	0.399	0.818	0.821
tinba	0.702	0.671	0.713	0.532	0.284	0.773	0.802
tempedreve	0.694	0.676	0.713	0.539	0.286	0.791	0.795
kraken_v1	0.771	0.746	0.757	0.775	0.693	0.758	0.790
necurs	0.648	0.616	0.782	0.439	0.259	0.816	0.781
ranbyus_v2	0.761	0.727	0.734	0.566	0.446	0.799	0.729
locky	0.588	0.578	0.601	0.446	0.375	0.633	0.622
bedep	0.615	0.606	0.603	0.513	0.489	0.622	0.619
kraken_v2	0.544	0.589	0.556	0.555	0.495	0.606	0.602
proslikefan	0.628	0.615	0.612	0.512	0.502	0.642	0.599
qakbot	0.603	0.592	0.548	0.448	0.292	0.602	0.542
cryptolocker	0.553	0.509	0.494	0.400	0.295	0.565	0.535
alureon	0.479	0.338	0.330	0.289	0.299	0.378	0.476
fobber_v2	0.302	0.277	0.422	0.348	0.192	0.467	0.366
ramnit	0.395	0.394	0.329	0.231	0.183	0.416	0.363
pykspa	0.357	0.346	0.392	0.309	0.210	0.396	0.359
pykspa_noise	0.303	0.292	0.229	0.239	0.264	0.329	0.321
dircrypt	0.353	0.302	0.326	0.312	0.294	0.359	0.303
accuracy	0.783	0.785	0.795	0.751	0.706	0.839	0.833
macro avg.	0.779	0.777	0.792	0.752	0.697	0.839	0.831
weighted avg.	0.779	0.777	0.792	0.752	0.697	0.839	0.831

b) UMUDGA T-540

FIGURE 7. Results of the multi-class classification over the UMUDGA T-17 (a) and T-540 (b). We report F1 scores for each DGA class obtained with the single models and with the two ensemble models, Stacked and Multi-input. We also report, in the last three rows, the accuracy and the macro and weighted average F1 score.

Furthermore, results on T-17 show that some challenging DGAs, like *sisron*, *vawtrak_v2* and *vawtrak_v3*, *pushdo*, *nymaim* and *gozi_luther*, where the random baseline scored less than 0.35 in F1, are very well detected by the ensemble models (F1 between 0.79 and 0.94).

Single models do not provide a uniform advantage over the different kinds of DGAs. For example, 1-grams performs better on *chinad* and *fobber_v1*, while 3-grams on *banjori* and *simmy*. Unsurprisingly, the word based model (ELMO) does well in detecting some dictionary-based DGAs, like *gozi_gpl* and *suppobox_3*.

2) FALSE POSITIVE RATE

In Figure 8, we compare the FPR obtained with the different models on the smallest and the biggest training sets, T-17 and T-540. Results clearly show that mixed models sensibly

mitigates the problem of false positives, which, however, remain an issue in a few-shot learning scenario (T-17). We also note that the Multi-input model slightly outperform the Stacked model.

To analyse the contribution of different DGAs, as anticipated in Section IV-C, we breakdown the FPR over the 50 different DGA classes. In Figure 9, we compare the FPR obtained on the T-540 dataset by the Multi-input model and by the randomly initialized LSTM model. For brevity, only classes with FPR greater than zero are included, i.e. 33 out of 50. We observe that, with enough training examples to learn from, the Multi-input model sensibly reduces the FPR for most of the DGAs, especially for those that produce a high FPR value using the random init. model, as *vawtrak_v2* and *v3*, *simda* and *suppobox_2*.

There are however a number of variants that still generate a relatively high number of false positives even on the T-540 dataset, as *nymaim*, *matsnu* and *suppobox_2*, even if the FPR is lower than the baseline.

In Table 4 examples of false positive domains, that are classified by the Multi-input model as nymaim, matsnu, suppobox_2 and vawtrak_v3 (the DGAs with the highest FPR), are reported.

We can observe that some of the false positives in the table exhibit uncommon word combinations, meaning they contain word sequences that are unusual in natural language or do not commonly appear together. Examples include *evertherenous.info*, *spotscenered.info*, *trustednotice.news*, *newsspeaker.com*, *notify-service.com* and *baymaleti.net*. None of these domains currently resolve to an active website.

However, several false positive domain names belong to well-known websites, such as *elmundo.es* (Spanish news site), *repubblica.it* (Italian news site), *rottentomatoes.com* (movie reviews), *stackoverflow.com* (Q&A site for programmers) and *wildberries.ru* (a large Russian e-commerce platform). This is a limitation of the contextless system analyzed in this study, although in real-world operational settings a detection system might also rely on white-lists - and on contextual information (e.g., WHOIS data, DNS resolution behaviour, traffic patterns) - to avoid blocking well-known websites.

In some cases, the misclassification of domains could be due to the presence of DGA domains with a similar n-gram pattern. For example, four nymaim domains in the dataset include the character sequence ''repub'' (i.e., *republicanopposite.la*, *techrepublic-modify.id*, *techrepubliceducation.ki* and *warepublishers.bt*), while the same sequence appears only once in the legitimate domain set. This could potentially cause the legitimate domain *repubblica.it* to be classified as nymaim. Similarly, the sequence ''plus'' appears in three nymaim domains (*surplusmotivation.mo*, *trackedplus.sg*, *vulnerable-plus.ec*) and only once in the legitimate dataset. This may have led to the legitimate domain *pelisplus.co* being mistakenly classified as *nymaim*. Some domains erroneously classified as matsnu include the ''-'' character to separate words. This pattern is frequently used by the matsnu DGA (in 886 out of 1,000 domains), potentially leading to some misclassifications.

3) FALSE NEGATIVE RATE

Another common issue in DGA detection systems is related to the FNR, which indicates the fraction of DGA domains that are recognized as legitimate, thus risking to bypass detection. This is particularly relevant in high risk situation, for example where the priority is to block an already active botnet. In Figure 10 we report the FNR obtained by the evaluated models for the T-17 dataset (the smallest) and T-540 (the largest).

Figure 10 a) reveals that, in a few shot learning condition, the baseline model and the 1-grams pre-trained model provides the lowest FNR. We recall, however (see Figure 6) that they provided a very low classification accuracy. On the other hand, pre-trained 3-grams and words embeddings, which provides accuracy advantages, especially on dictionary-based DGAs, do it at the cost of a greater FNR value. The effect is somehow mitigated by the Multi-input model, which provides a FNR of 0.0048, even if still above the baseline.

As the number of training examples increases, the trend is inverted in the T-540 dataset (see Figure 10 (b)), where the mean FNR of the Multi-input model is the lowest among all models, and less than an half of the value registered for the random LSTM baseline and for the word-based model, while, at the same time, providing higher classification accuracy (see Figure 6). Once again, the Multi-input model, that registers a FNR of 0.0017, outperforms the Stacked model.We also note that the FNR provided by the word-based model is almost the same as in T-17, while in the case of the 2-grams model the FNR is higher in T-540.

Figure 11 shows the FNR breakdown for each single DGAs. 19 out of 50 DGAs show a FNR equal to zero; they have been removed from the figure. Overall, the FNR provided by the Multi-input model is sensibly lower that the one provided by the baseline for almost all DGAs. An exception is given by *matsnu* and *gozi_rfc4343* where, however, the difference with the baseline is small. On the contrary, the DGAs with a high baseline FNR, like *simda*, *vawtrak_v3* and others, are handled very well by the Multi-input model, reducing the FPR to nearly zero.

Results show how specific DGAs are more challenging, as they have a higher chance of being confused with legitimate DNs, even if more training example are added.

Among them, *nymaim* is the most effective dictionary based DGA, and provides the highest FPR (0.029) and FNR (0.016), followed by *matsnu* (FPR = 0.013, FNR = 0.007). Both the DGAs are based on the concatenation of two words coming from pre-defined distinct word-lists. Other dictionary based DGAs are generally better detected, i.e. the variants of *gozi*, which derive DNs from web-accessible documents such as the US constitution or RFCs.

Results also show that, beside dictionary based DGAs, that are generally recognized to be challenging in literature, other, more traditional kinds of DGAs still can represent

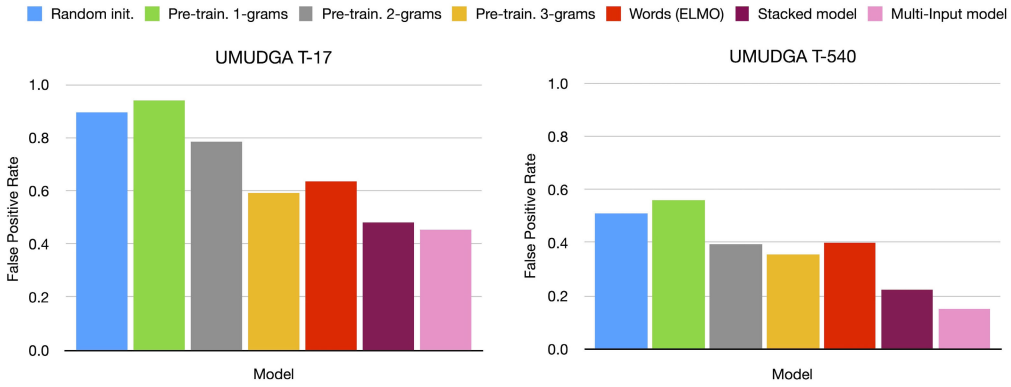

FIGURE 8. False Positive Rate(FPR) obtained on the T-17 and T-540 UMUDGA subset with different models.

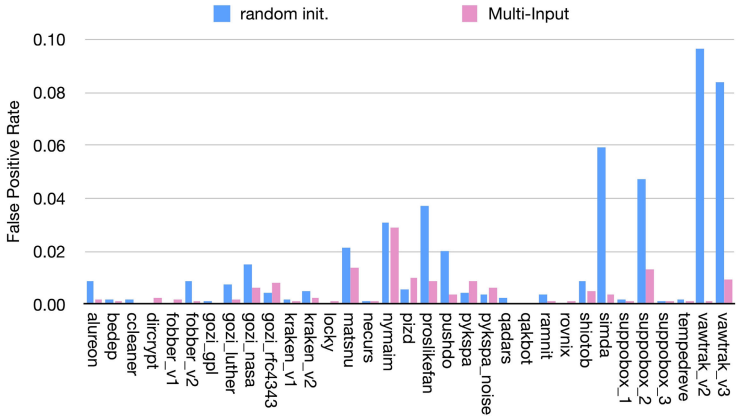

FIGURE 9. FPR breakdown by DGA class in UMUDGA T-540.

TABLE 4. False Positive (FP) domain examples for the 4 DGA families with higher FPR.

DGA	FPR	FP domains examples
nymaim	0.029	apparedistride.club, archiveofourown.org, biobiochile.cl, elmundo.es, evertherenous.info, freeadult.games, furaffinity.net, kissasian.sh, pelisplus.co, pirateproxy.app, playground.ru, repubblica.it, rottentomatoes.com, slickdeals.net, spotscenered.info, tamilrockers.to, trustednotice.news, wildberries.ru, yourporn.sexy
matsnu	0.013	list-manage.com, news-speaker.com, notify-service.com, pathofexile.com, perfecttool-media.com, premierleague.com, segmentfault.com, stackoverflow.com, steamcommu-nity.com, steampowered.com, ticketmaster.com, ultimate-guitar.com, wordreference.com
suppobox_2	0.013	baymaleti.net, cloudfront.net, ettoday.net, popcash.net, readms.net, savefrom.net, slick-deals.net, speedtest.net, studfiles.net, tebyan.net
vawtrak_v3	0.010	analdin.com, bongacams.com, gofundme.com, gosuslugi.ru, lifewire.com, namasha.com, ninisite.com, softonic.com, usatoday.com, velocecdn.com, wellsfargo.com, zillow.com

TABLE 5. False Negative (FN) domain examples for the 4 DGA families with higher FNR.

DGA	FNR	FN domains examples
nymaim	0.016	sideseaster.com, skirtebay.am, sortlyric.kr, stemcove.am, streetmaria.net, tankdisney.com, thanksfred.com, tobagojazz.co, ultrapolitics.com, voluntaryhello.com, waitemma.co, walesjack.com, wannalaunch.com, whitegothic.co, worm-pain.com
pykspa_noise	0.011	apatretc.info, bakuvet.com, elogsu.net, henalskytia.com, nelperogp.com, nghtondo.net, qusaso.org, rnraredak.org
proslikefan	0.010	celstrpick.org, curcuq.in, etomzap.com, futess.name, imathaxi.net, kujaky.org, ororbx.net, pcavnd.se, shidindfd.com, uissote.net, vvvrern.com
matsnu	0.007	assisttravel.com, databasecoast.com, goalchampion.com, manufacturer.com, monthat-tempt.com, officerweigh.com, professorblue.com, video-express.com

treats. *Proslikefan*, for example has a FPR of 0.009 and a FNR of 0.01, followed by two variants of the same DGA, *pykspa_noise* (FPR = 0.006 and FNR = 0.011), *pykspa* (FPR = 0.0086 and FNR = 0.0047). Both the

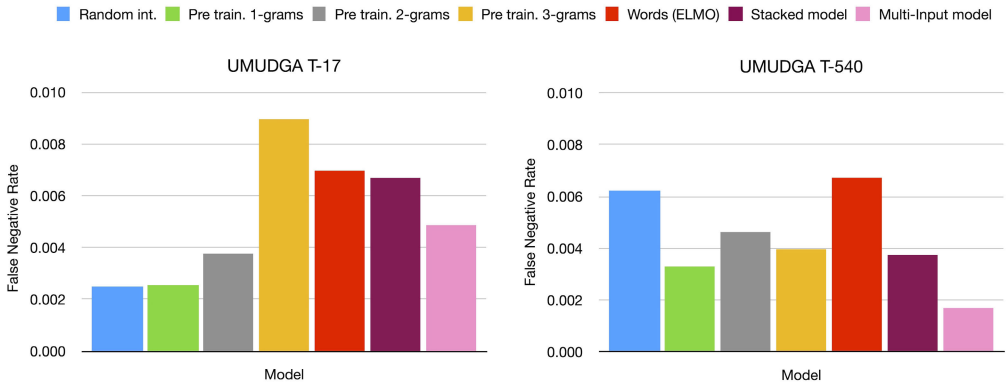

FIGURE 10. False Negative Rate (FNR) obtained on the T-17 and T-540 UMUDGA subset with different models.

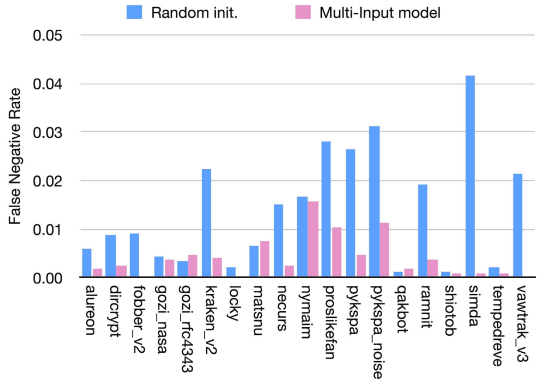

FIGURE 11. FNR breakdown by DGA class in UMUDGA T-540.

algorithms outputs medium length DNs (6 to 12 charts) based on a numerical calculation from pseudo random seeds. Furthermore, we observe that, while *nymaim* and *matsnu* have high F1-scores, over 0.94, *pykspa* and *proslikefan* has low F1 scores. This means that, while the dictionary-based DGAs are clearly distinguishable from other DGAs, the char-based ones can be more difficult to identify in a multi-class classification scenario.

In Table 5, examples of false negative domain names of the four DGAs with higher FNR (nymaim, pykspa_noise, proslikefan and matsnu) are reported. While in the case of vocabulary-based DGAs (nymaim and matsnu) the false negatives are probably due to domain names resembling legit domains, finding a rationale for the false negatives of pykspa_noise and proslikefan is more complex. A first aspect that certainly needs to be considered is the balance between vowels and consonants in several domain names generated by these two DGAs (see *kujaky.org* and *ororbx.net* for proslikefan, *elogsu.net*, *qusaso.org* for pykspa_noise), which contrasts with the use of semi-random vowel/consonant patterns used by the DGAs. A second aspect is the presence of character sequences that phonetically recall real words and allude to human generated domains, such as *celstrpick.org* (cell, string, pick) and *imathaxi.net* (a Greek sounding stucture) for proslikefan, *bakuvet.com* (baku, vet) and *henalskytia.com* (henal, Spanish term for hayloft, and sky).

C. MIXED MODELS COMPARISON

Finally, let us focus on comparing the two ensemble strategies considered in this study: one based on a more traditional technique, the stacked model, and the other were fusion is performed by a multi-channel deep neural network, the Multi-input model.

We can say that performances in term of accuracy and F1 score in a DGA multi-class classification task prove to be very similar: they both absorb the best from the composing models, outperforming them. Results from the 25-DGA also suggested that the Multi-input model better capture legitimate domains and dictionary based DGAs in few-shot conditions. However, such results are not clearly observed, with a larger DGAs number, in the UMUDGA dataset.

On the other hand, while both prove lowest FPR and FNR compared to single models, we observe (see Figures 8 and 10) that the Multi-input model clearly outperforms the stacked model in robustness to false positive and negative predictions.

In Table 6 and Table 7, we compare the two ensemble models, Stacked and Multi-input, in terms of F1 gain with respect to the F1 score achieved by the best performing single model within the ensemble. Along with the average gain across all DGAs, we report the number of DGAs in which the ensemble underperforms (yielding a lower F1 score than the best single model), the average ΔF1 in these cases (a negative value, i.e. the loss) and the same figures for DGAs where the ensemble outperforms. Finally, to highlight relevant differences in F1 loss and gain, we defined a threshold value of 0.05 (i.e. 5% change in F1) and we included in the tables the number of DGAs where the F1 gain exceeds (> 0.05) or the loss is above this threshold (< -0.05). We conduct this analysis for both the 25-DGA and UMUDGA datasets, considering results from the smallest sub-dataset (T-17) and the largest one (T-540).

Although the ensemble models generally improves the overall F1 score compared to individual models (as

TABLE 6. Comparison between ensemble models and best performing single model on 25-DGA T-17 and T-540 datasets.

	T-17		T-540	
	Stacked	Multi-input	Stacked	Multi-input
Avg. ΔF1	0.006	**0.067**	0.026	**0.035**
# DGAs with F1 loss	9	**0**	**0**	5
Avg. F1 loss	-0.058	**0.000**	**0.000**	-0.010
# DGAs with F1 gain	16	**25**	**24**	20
Avg. F1 gain	0.043	**0.062**	0.029	**0.032**
# DGAs with F1 loss < -0.05	4	**0**	0	0
# DGAs with F1 gain > 0.05	3	**11**	3	**4**

TABLE 7. Comparison between ensemble models and best performing single model on UMUDGA T-17 and T-540 datasets.

	T-17		T-540	
	Stacked	Multi-input	Stacked	Multi-input
Avg. ΔF1	0.029	**0.059**	**0.016**	0.008
# DGAs with F1 loss	15	**6**	**4**	13
Avg. F1 loss	**-0.053**	-0.055	-0.030	**-0.028**
# DGAs with F1 gain	35	**44**	**42**	30
Avg. F1 gain	0.064	**0.075**	0.022	**0.025**
# DGAs with F1 loss < -0.05	7	**3**	**1**	2
# DGAs with F1 gain > 0.05	20	**29**	5	**7**

previously discussed), in some DGAs they still exhibit lower performance, indicating room for improvement. Notably, in the worst case, the Stacked ensemble underperforms in 36% of DGAs in the 25-DGA T-17 dataset (9 out of 25) and in 30% of DGAs (15 out of 50) in the UMUDGA T-17 dataset, while considering losses greater than 0.05 the Stacked model underperforms in 16% and 14% of cases, respectively.

In the T-17 setting of both datasets (25-DGA and UMUDGA), the Multi-input model achieves a higher average F1 gain compared to the Stacked model (0.067 vs. 0.006 in 25-DGA and 0.059 vs. 0.029 in UMUDGA) while in T-540 setting this model does better in the 25-DGA dataset and worse in the UMUDGA dataset, thus confirming that it is better suited for low-data scenarios, but as the dataset grows the two models tend to have similar performances.

In 25-DGA T-17 dataset, the Stacked model underperforms on 9 DGAs, while the Multi-input model does not exhibit an F1 loss for any DGA. Similarly, in UMUDGA T-17 dataset, the Multi-input model results in only 6 DGAs with negative F1, compared to 15 for the Stacked model. The situation is reversed in the T-540 case where, in both datasets, the Multi-input has more DGAs with performance drop (5 vs. 0 in 25-DGA, 13 vs. 4 in UMUDGA).

A similar situation appears if we consider the number of DGAs in which the ensembles shows a gain in F1. There are more DGAs in which Multi-input gains with respect to Stacked in the 25-DGA T-17 datasets (25 vs. 16) and UMUDGA T-17 (44 vs. 35), fewer in the 25-DGA T-540 (20 vs. 14) and UMUDGA T-540 (30 vs. 42). As to the magnitude of the average F1 loss, we see that in two cases the Multi-input loses more than the Stacked (25-DGA T-540 and UMUDGA T-17), in two less (25-DGA T-17 and UMUDGA T-540), and that the significantly greater loss, -0.058, is for the Stacked model in the 25-DGA T-17 dataset. The situation for the average F1 gain is different: in all datasets the Multi-input does better than the Stacked, with a maximum difference of 0.019 in the 25-DGA T-17 dataset.

The number of DGAs with an F1 gain > 0.05 is consistently higher for Multi-input across both datasets and training sizes. 25-DGA T-17: 11 DGAs benefit from an F1 gain > 0.05 with Multi-input, while only 3 do with Stacked. UMUDGA T-17: 29 DGAs benefit with Multi-input, compared to 20 with Stacked. Even in T-540, Multi-input outperforms Stacked in this aspect (4 vs. 3 for 25-DGA and 7 vs. 5 for UMUDGA). This further confirms that Multi-input exhibits stronger synergy within the ensemble, particularly in low-data scenarios (T-17).

Overall, the data show that Multi-input ensemble exhibits stronger synergy among its components. Indeed, it has an average gain in F1 to the best performing single model that overcome the one of the Stacked ensemble in three of the four dataset used (25-DGA T-17, 25-DGA T-540 and UMUDGA T-17). It is the preferred ensemble model, especially when training data is limited. Going into detail about its performance on single DGAs, if we focus on those with severe F1 loss and we consider the UMUDGA T-17 dataset, three DGAs, namely *gozi_rfc4343*, *alureon* and *fobber_v2*, have a loss of around -0.08. The situation changes in the UMUDGA T-540 dataset, where the gain becomes positive for *gozi_rfc4343* (+0.09), the loss is reduced for *alureon* (-0.003) but remains significant for *fobber_v2* (-0.056). This shows how the classifier, regardless of the size of the training dataset, has difficulty recognizing a limited number of DGAs.

Finally, the two mixed models differ in the training process. While the stacked model needs a multi phase training, with additional data splits, the Multi-input model is trained end-to-end, therefore is faster and simpler to train.

We measured the training and prediction times of the models on Google Colab on a machine equipped with a

TABLE 8. Summary of training and testing times per single domain measured for the ensemble models and for single components.

Ensemble models		
	Training	Prediction
Multi-Input	26 ms.	10 ms.
Stacked	39 ms.	10 ms.

Single components		
	Training	Prediction
n-grams	1 ms.	0.3 ms.
words	19 ms.	7 ms.

NVIDIA Tesla T4 GPU (16 GB GDDR6 VRAM) and 32 GB of RAM. Results are summarized in Table 8.

Training the Multi-input model took approximately 26 milliseconds per data sample, while predicting a single domain name required around 10 milliseconds.

Among the individual components, the most computationally intensive is the one handling word embeddings, which took approximately 19 milliseconds per domain during training and around 7 milliseconds per domain during prediction. The other components (1-grams with random embeddings,, 1-grams, 2-grams, and 3-grams with pre-trained embeddings) showed minimal variation, requiring about 1 millisecond per sample during training and 0.3 milliseconds per sample at prediction time.

The Stacked model requires separate training for each of the five channels in the first stage, followed by an additional training phase for the final stage. The computational effort for training the first stage equals the sum of the efforts required to train all the individual components. Assuming sequential training, the total training time amounts to approximately 23 milliseconds per data point.

In the final stage, a logistic regression model integrates the outputs from the five channels. During this phase, the LSTM-based components are used only for forward propagation (prediction), while the logistic regression module undergoes training. This process took approximately 40 milliseconds per data point.

At inference time, the Stacked model achieves prediction times that are practically equivalent to those of the Multi-input model, requiring around 10 milliseconds per predicted domain.

D. ABLATION STUDY

To further investigate the impact of each domain name tokenization (e.g. 2-grams, 3-grams, words) and the corresponding pre-trained embeddings, we conducted an ablation study, by removing each component from the multi-input ensemble model and averaging the results over 10 folds. Results obtained, in terms of overall classification accuracy, on the UMUDGA T-17 and T-540 are summarized in Figure 12. In the figure, we also show the accuracy provided by the best of the single models, which happens to be the 2-grams based model.

As one can see, the removal of 1-grams (both with randomly initialised and with pre-trained embeddings) results in a very small decrease in accuracy, both in the case of the smallest dataset (T-17) and of the largest (T-540). While word embeddings are the most important feature with few training examples(i.e. T-17), its importance decreases with the increase of training examples (T-540). On the contrary, the impact of 2-grams seems to grow with the increase of training examples: in T-540 removing 2-grams causes the largest accuracy drop. We notice, however, that accuracy drop in T540 is quite limited in general (around 1% maximum).

Surprisingly, removing 3-grams, which resulted in a relatively large accuracy drop in T-17, provides a small accuracy improvement (around 0-2%) in T-540. While the gap is not significant, this confirms that, while 3-grams helps in capturing meaningful sequences in a few shot setting, they become less important with the growth of the training examples. Overall, the results of the ablation experiment confirm the trends observed in Section V-B.

Finally, we notice that, as the word embedding matrix is by far the largest, the use of words in the ensemble results in a much more expensive training phase (approximately 10 times longer in our experiments). While this may not be an issue with small datasets, it could pose a challenge with larger ones, especially considering that, in real-world settings, detectors require constant retraining to cope with new DGAs. However, removing word embeddings from T-540 didn't result in a significant performance drop, making it a viable option to address computational constraints.

TABLE 9. Multi-class classification accuracy.

Model	Overall accuracy
Multi-Input (Ours)	**0.863**
LA_Mul07 [68]	0.859
LSTM.MI [19]	0.854
BiLSTM	0.855
A-CNN-BLSTM	0.856

E. COMPARISON WITH STATE OF THE ART METHODS

In this section we provide an overall assessment of the system comparing it with existing deep learning methods for Multi-class classification of DGA families. Overall accuracy obtained on the full UMUDGA dataset (10K tier) are shown in Table 9. We compared our Multi-Input model with the LA_Mul07, based on specialized attention mechanism, proposed in [68], LSTM.MI [19], A-CNN-BLSTM [49], that use a hybrid CNN/LSTM architecture and BiLSTM [54].

As shown, the results obtained by our Multi-level model over the 10K UMUDGA dataset is competitive and beats the best performing among the existing methods (LAMul07), in overall multi-class classification accuracy.

To further analyse the performance of the Multi-input model against the others, in Table 10, we show the F1 score gain achieved by the Multi-input model in comparison with existing approaches on the 10K UMUDGA dataset. Specifically, we report the average F1 gain, the number of DGAs where performance decreases (negative gain), the magnitude of negative and positive gains, and the number of

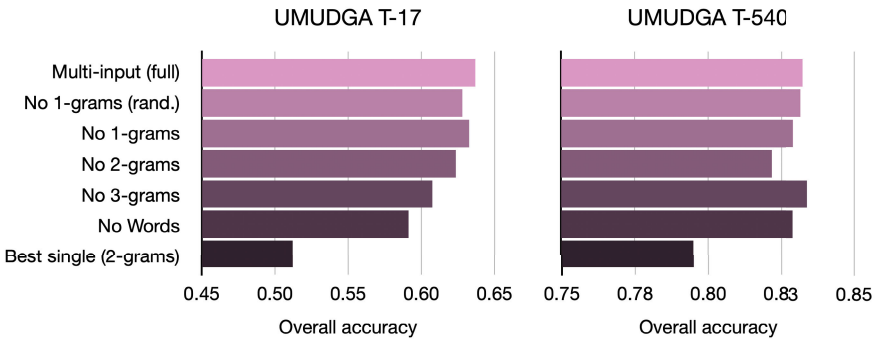

FIGURE 12. Overall accuracy variation on the UMUDGA T-17 and T-540 dataset, by removing models from the multi-input ensemble.

TABLE 10. Analysis of F1 score obtained with the Multi-input model vs. existing approaches on full UMUDGA dataset.

	vs. LAMul07	vs. A-CNN-BLSTM	vs. BLSTM	vs. LSTM.MI
Avg. ΔF1	0.008	0.114	0.122	0.128
# DGAs with F1 loss	25	2	4	1
Avg. F1 loss	-0.0204	-0.0002	-0.0003	-0.0001
# DGAs with F1 gain	20	44	41	45
Avg. F1 gain	0.043	0.126	0.146	0.139
# DGAs with F1 loss < -0.05	3	0	0	0
# DGAs with F1 gain > 0.05	6	24	23	25

DGAs with significant improvement (>0.05) or degradation (<-0.05).

The average F1 gain increases progressively from 0.01 (vs. LAMul07) to 0.13 (vs. LSTM.MI), confirming that LAMul07 is the most competitive baseline. While F1 loss is limited to 4, 2 and 1 DGAs in the cases of, respectively, BiLSTM, A-CNN-BLSTM and LSTM.MI, there are 25 DGAs (50%) where the Multi-input model underperforms LAMul07. Similarly, the Multi-input model has a better F1 for 45 DGAs compared to LSTM.MI, 44 compared to A-CNN-BLSTM and only 20 compared to LaMul07, on which it maintains, as mentioned, a slightly higher average ΔF1.

The average F1 loss is very small (near zero) for all models except LAMul07, where the drop is -0.0204. However, the average F1 gain (0.043) is notably higher, and the number of DGAs experiencing a significant loss (3) is lower than those exhibiting a significant gain (6).

F. LIMITS

While accuracy in the multiclass classification is generally good, as happens with other deep learning methods [68] some families of DGAs have low metrics, potentially posing serious challenges when considering real world detection systems. While experiments in this paper was done relying on available datasets, we acknowledge such dataset have limitations. For example, the set of legitimate domains includes a limited number of randomly generated domains and short domains and no domains with hash values, which are usually challenging for contextless detection systems. Little effort has been put, to the best of our knowledge, to create more realistic datasets resembling real world scenarios, e.g. including HTTP calls to enterprise services of applications, where domains potentially include pseudo random ids or alphanumeric strings. Following this direction, we are planning to experiment with simulated HTTP calls to existing enterprise APIs. Moreover, while a quantitative analysis on the false positives produced by the classifier has been conducted, our study does not delve into an analysis of potential weaknesses in the detection method or the ways in which adversaries might evade detection. These aspects require further investigation.

VI. CONCLUSION AND FUTURE WORKS

This work deals with the multi-class DGA classification with small training datasets. The basis of this work are three research questions, reported in Section I, and the idea of investigating the use of different lexical features extracted from DNs to pre-train a Deep Learning model with unsupervised embeddings. After the experiments whose results have been reported and discussed in Section V, the following conclusions can be drawn.

Regarding Research Questions RQ1 and RQ2, we designed a novel LSTM-based ensemble models to fuse the prediction ability of different DNs representations, namely n-grams and words series, and we conducted extensive evaluation on two different datasets. Overall, our results are encouraging if we consider that feature based methods and deep learning methods accuracy currently ranges between 77% and 89% [6] on multi-class classification using the whole available 10k tier UMUDGA training set, while the results of the mixed models, in the presented limited training size conditions (T-540) reaches 84%. The ensemble models are shown to increase accuracy, both with respect to the LSTM baseline and to single pre-trained models, over all the analyzed

training size range, from few shot learning to medium size datasets. Furthermore, the multi-input model, based on hidden representation fusion in a multi-layer neural network, provides low FPR and FNR, thus improving robustness in practical applications.

Regarding Research Question RQ3, the approach followed was to work at the feature level FSL, according to the taxonomy proposed by Song et al. [32]. In particular, in this work we worked in the pre-training stage, using both n-gram embedding and word embedding techniques. While the improvements provided by our method in few-shot conditions are evident, results still indicate margins of improvements and show that some challenging DGAs produce DNs that are hard to distinguish from legitimate ones. While in literature dictionary based DGAs are indicated as the most challenging, our results shows that it is not always the case. We believe that further investigating the use of unsupervised learning to tackle this issues is a promising direction.

In future works, several improvement to the framework proposed in this study could be investigated. For example, regarding word based embedding, the robustness against dictionary-based DGAs could be improved by integrating multi-language or language-independent neural language models. Furthermore, additional experiments could be conducted to assess various corpora of domain names for learning token embeddings, considering variations in dataset size and in domain names type (e.g. already known DGAs, less popular web sites).

In addition, while in this study we built our single models on recurrent LSTM network, evaluating and tuning different architectures, such as integrating CNNs or attention mechanisms could be investigated to further improve performances.

Finally, since the primary focus of this paper is on the conceptual definition of an original model for DGA detection, deployment feasibility aspects such as latency, throughput, and zero-shot adaptability will be investigated in future work.

REFERENCES

[1] SophosLab. (2023). *Sophos 2023 Threat Report-Maturing Criminal Marketplaces Present New Challenges To Defenders*. [Online]. Available: https://assets.sophos.com/X24WTUEQ/at/b5n9ntjqmbkb8fg5rn25g4fc/sophos-2023-threat-report.pdf

[2] Spamhaus. (2023). *Spamhaus Botnet Threat Update: Q2-2023*. [Online]. Available: https://info.spamhaus.com/hubfs/Botnet%20Reports/2023%20Q2%20Botnet%20Threat%20Update.pdf? hsCtaTracking=587a97f0-87c7-4afd-a33d-6118773af723%7Cb52560a5-44e0-4cae-9113-762047f9f6fd

[3] Eur. Union Agency for Cybersecurity. (2022). *ENISA Threat Landscape 2022*. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat- landscape-2022/@@download/fullReport

[4] D. Plohmann, K. Yakdan, M. A. Klatt, J. Bader, and E. Gerhards-Padilla, "A comprehensive measurement study of domain generating malware," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 263–278.

[5] A. Drichel, B. Holmes, J. von Brandt, and U. Meyer, "The more, the better: A study on collaborative machine learning for dga detection," in *Proc. 3rd Workshop Cyber-Secur. Arms Race*, New York, NY, USA, Nov. 2021, pp. 1–12.

[6] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, and M. Baldi, "Algorithmically generated malicious domain names detection based on n-grams features," *Expert Syst. Appl.*, vol. 170, May 2021, Art. no. 114551.

[7] B. Yu, J. Pan, J. Hu, A. C. A. Nascimento, and M. D. Cock, "Character level based detection of DGA domain names," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[8] A. Drichel, U. Meyer, S. Schüppen, and D. Teubert, "Analyzing the real-world applicability of DGA classifiers," in *Proc. 15th Int. Conf. Availability, Rel. Secur.*, Aug. 2020, pp. 1–11.

[9] H. Suryotrisongko, Y. Musashi, A. Tsuneda, and K. Sugitani, "Robust botnet DGA detection: Blending XAI and OSINT for cyber threat intelligence sharing," *IEEE Access*, vol. 10, pp. 34613–34624, 2022.

[10] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, p. 14, 2014.

[11] S. Schiavoni, F. Maggi, L. Cavallaro, and S. Zanero, "Phoenix: DGA-based botnet tracking and intelligence," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, S. Dietrich, Ed., Cham, Switzerland: Springer, 2014, pp. 192–211.

[12] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Process. Lett.*, vol. 48, no. 3, pp. 1347–1357, Dec. 2018.

[13] R. Sivaguru, J. Peck, F. Olumofin, A. Nascimento, and M. De Cock, "Inline detection of DGA domains using side information," *IEEE Access*, vol. 8, pp. 141910–141922, 2020.

[14] M. Zago, M. G. Pérez, and G. Martínez Pérez, "Scalable detection of botnets based on DGA: Efficient feature discovery process in machine learning techniques," *Soft Comput.*, vol. 24, no. 8, pp. 1–21, Jan. 2019.

[15] A. Singla and E. Bertino, "How deep learning is making information security more intelligent," *IEEE Secur. Privacy*, vol. 17, no. 3, pp. 56–65, May 2019.

[16] V. Ravi, M. Alazab, S. Srinivasan, A. Arunachalam, and K. P. Soman, "Adversarial defense: DGA-based botnets and DNS homographs detection through integrated deep learning," *IEEE Trans. Eng. Manag.*, vol. 70, no. 1, pp. 249–266, Jan. 2023.

[17] K. Highnam, D. Puzio, S. Luo, and N. R. Jennings, "Real-time detection of dictionary DGA network traffic using deep learning," *Social Netw. Comput. Sci.*, vol. 2, no. 2, p. 110, Apr. 2021.

[18] H. Shahzad, A. Sattar, and J. Skandaraniyam, "DGA domain detection using deep learning," in *Proc. IEEE 5th Int. Conf. Cryptography, Secur. Privacy (CSP)*, Jan. 2021, pp. 139–143.

[19] 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, Jan. 2018.

[20] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," in *Proc. 21st USENIX Conf. Secur. Symp.*, Aug. 2012, pp. 491–506.

[21] G. Wu and E. Y. Chang, "Adaptive feature-space conformal transformation for imbalanced-data learning," in *Proc., 20th Int. Conf. Mach. Learn.*, Aug. 2003, pp. 816–823.

[22] X. Pei, S. Tian, L. Yu, H. Wang, and Y. Peng, "A two-stream network based on capsule networks and sliced recurrent neural networks for DGA botnet detection," *J. Netw. Syst. Manage.*, vol. 28, no. 4, pp. 1694–1721, Oct. 2020.

[23] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. C. A. Nascimento, and M. De Cock, "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51542–51556, 2019.

[24] M. Zago, M. Gil Pérez, and G. Martínez Pérez, "UMUDGA: A dataset for profiling DGA-based botnet," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101719.

[25] A. J. Aviv and A. Haeberlen, "Challenges in experimenting with botnet detection systems," in *Proc. 4th Workshop Cyber Secur. Experimentation Test*, Aug. 2011, p. 6.

[26] A. Drichel, U. Meyer, S. Schüppen, and D. Teubert, "Making use of NXt to nothing: The effect of class imbalances on DGA detection classifiers," in *Proc. 15th Int. Conf. Availability, Rel. Secur.*, New York, NY, USA, Aug. 2020, pp. 1–9, doi: 10.1145/3407023.3409190.

[27] S. Skuratovich. (2015). *Matsnu*. Check Point Softw. Technol. [Online]. Available: https://blog.checkpoint.com/wp-content/uploads/2015/07/matsnu-malwareid-technical-brief.pdf

[28] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.

[29] Z. Tang, P. Wang, and J. Wang, "ConvProtoNet: Deep prototype induction towards better class representation for few-shot malware classification," *Appl. Sci.*, vol. 10, no. 8, p. 2847, Apr. 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/8/2847

[30] C. Xu, J. Shen, and X. Du, "Detection method of domain names generated by DGAs based on semantic representation and deep neural network," *Comput. Secur.*, vol. 85, pp. 77–88, Aug. 2019.

[31] J. J. Koh and B. Rhodes, "Inline detection of domain generation algorithms with context-sensitive word embeddings," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 2966–2971.

[32] Y. Song, T. Wang, P. Cai, S. K. Mondal, and J. P. Sahoo, "A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–40, Dec. 2023.

[33] C. Morbidoni, L. Spalazzi, A. Teti, and A. Cucchiarelli, "Leveraging n-gram neural embeddings to improve deep learning DGA detection," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2022, pp. 995–1004.

[34] W. Huang, Y. Zong, Z. Shi, L. Wang, and P. Liu, "PEPC: A deep parallel convolutional neural network model with pre-trained embeddings for DGA detection," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.

[35] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, p. 1249, Jul. 2018.

[36] P. M. da Luz, "Botnet detection using passive DNS," Master's thesis, Dept. Comput. Sci., Radboud Univ., Nijmegen, The Netherlands, 2014.

[37] A. Ahluwalia, I. Traoré, K. Ganame, and N. Agarwal, "Detecting broad length algorithmically generated domains," in *Proc. Int. Conf. Intell., Secure, Dependable Syst. Distrib. Cloud Environ.*, Jan. 2017, pp. 19–34.

[38] Y. Li, K. Xiong, T. Chin, and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection," *IEEE Access*, vol. 7, pp. 32765–32782, 2019.

[39] J. Selvi, R. J. Rodríguez, and E. Soria-Olivas, "Detection of algorithmically generated malicious domain names using masked N-grams," *Expert Syst. Appl.*, vol. 124, pp. 156–163, Jun. 2019.

[40] F. Casino, N. Lykousas, I. Homoliak, C. Patsakis, and J. Hernandez-Castro, "Intercepting hail hydra: Real-time detection of algorithmically generated domains," *J. Netw. Comput. Appl.*, vol. 190, Sep. 2021, Art. no. 103135. [Online]. Available: https://www.sciencedirect.com/science/article/pii/

[41] C. Patsakis and F. Casino, "Exploiting statistical and structural features for the detection of domain generation algorithms," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102725.

[42] I. Chiscop, F. Soro, and P. Smith, "AI-based detection of DNS misuse for network security," in *Proc. 1st Int. Workshop Native Netw. Intell.*, New York, NY, USA, Dec. 2022, pp. 27–32.

[43] J. Liang, S. Chen, Z. Wei, S. Zhao, and W. Zhao, "HAGDetector: Heterogeneous DGA domain name detection model," *Comput. Secur.*, vol. 120, Sep. 2022, Art. no. 102803.

[44] N. Kostopoulos, D. Kalogeras, D. Pantazatos, M. Grammatikou, and V. Maglaris, "SHAP interpretations of tree and neural network DNS classifiers for analyzing DGA family characteristics," *IEEE Access*, vol. 11, pp. 61144–61160, 2023.

[45] 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 Trans. Netw.*, vol. 20, no. 5, pp. 1663–1677, Oct. 2012.

[46] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2017, pp. 1–14.

[47] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, Jan. 2016, pp. 13–21.

[48] B. Yu, D. L. Gray, J. Pan, M. D. Cock, and A. C. A. Nascimento, "Inline DGA detection with deep networks," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 683–692.

[49] F. Ren, Z. Jiang, X. Wang, and J. Liu, "A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network," *Cybersecurity*, vol. 3, no. 1, p. 4, Dec. 2020.

[50] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, "Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework," *IEEE Access*, vol. 8, pp. 82876–82889, 2020.

[51] S. R. C. Liew and N. F. Law, "Use of subword tokenization for domain generation algorithm classification," *Cybersecurity*, vol. 6, no. 1, p. 49, Sep. 2023.

[52] P. Lison and V. Mavroeidis, "Automatic detection of malware-generated domains with recurrent neural models," 2017, *arXiv:1709.07102*.

[53] D. Yuan, Y. Xiong, T. Zang, and J. Huang, "Nemesis: Detecting algorithmically generated domains with an LSTM language model," in *Collaborative Computing: Networking, Applications and Worksharing* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 292. Cham, Switzerland: Springer, 2019, pp. 350–363.

[54] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "DGA botnet detection using supervised learning methods," in *Proc. 8th Int. Symp. Inf. Commun. Technol.*, Dec. 2017, pp. 211–218, doi: 10.1145/3155133.3155166.

[55] Y. Liang, J. Deng, and B. Cui, "Bidirectional LSTM: An innovative approach for phishing URL identification," *Adv. Intell. Syst. Comput.*, vol. 994, pp. 326–337, Jun. 2019.

[56] C. Yang, T. Lu, S. Yan, J. Zhang, and X. Yu, "N-trans: Parallel detection algorithm for DGA domain names," *Future Internet*, vol. 14, no. 7, p. 209, Jul. 2022.

[57] L. Ding, P. Du, H. Hou, J. Zhang, D. Jin, and S. Ding, "Botnet DGA domain name classification using transformer network with hybrid embedding," *Big Data Res.*, vol. 33, Aug. 2023, Art. no. 100395.

[58] B. Gogoi and T. Ahmed, "DGA domain detection using pretrained character based transformer models," in *Proc. IEEE Guwahati Subsection Conf. (GCON)*, Jun. 2023, pp. 01–06.

[59] 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," *Appl. Sci.*, vol. 9, no. 20, p. 4205, Oct. 2019.

[60] L. Sidi, Y. Mirsky, A. Nadler, Y. Elovici, and A. Shabtai, "Helix: DGA domain embeddings for tracking and exploring botnets," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2020, pp. 2741–2748.

[61] X. Fang, X. Sun, J. Yang, and X. Liu, "Domain-embeddings based DGA detection with incremental training method," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2020, pp. 1–6.

[62] S. Lin, S. Zhong, and K. Cheng, "A method with pre-trained word vectors for detecting wordlist-based malicious domain names," *J. Phys., Conf. Ser.*, vol. 1757, no. 1, Jan. 2021, Art. no. 012171.

[63] L. Yang, G. Liu, W. Liu, H. Bai, J. Zhai, and Y. Dai, "Detecting multielement algorithmically generated domain names based on adaptive embedding model," *Secur. Commun. Netw.*, vol. 2021, pp. 1–20, May 2021.

[64] J. Huang, P. Wang, T. Zang, Q. Qian, Y. Wang, and M. Yu, "Detecting domain generation algorithms with convolutional neural language models," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng., Trustcom/BigDataSE*, Aug. 2018, pp. 1360–1367.

[65] H. Cheng, Y. Fang, L. Chen, and J. Cai, "Detecting domain generation algorithms based on reinforcement learning," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery, CyberC*, Oct. 2019, pp. 261–264.

[66] M. Yang, "A survey on few-shot learning in natural language processing," in *Proc. Int. Conf. Artif. Intell. Electromech. Autom. (AIEA)*, May 2021, pp. 294–297.

[67] X. Hu, M. Li, G. Cheng, R. Li, H. Wu, and J. Gong, "Towards accurate DGA detection based on Siamese network with insufficient training samples," in *Proc. ICC IEEE Int. Conf. Commun.*, May 2022, pp. 2670–2675.

[68] T. A. Tuan, H. V. Long, and D. Taniar, "On detecting and classifying DGA botnets and their families," *Comput. Secur.*, vol. 113, Feb. 2022, Art. no. 102549.

[69] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," 2013, *arXiv:1312.3005*.

[70] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*.

[71] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[72] D. S. Berman, "DGA CapsNet: 1D application of capsule networks to DGA detection," *Information*, vol. 10, no. 5, p. 157, Apr. 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/5/157

[73] R. Sivaguru, C. Choudhary, B. Yu, V. Tymchenko, A. Nascimento, and M. D. Cock, "An evaluation of DGA classifiers," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5058–5067.

[74] F. Bisio, S. Saeli, P. Lombardo, D. Bernardi, A. Perotti, and D. Massa, "Real-time behavioral DGA detection through machine learning," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2017, pp. 1–6.

CHRISTIAN MORBIDONI received the M.Sc. degree in electronic engineering and the Ph.D. degree in computer science from the Polytechnic University of Marche, in 2003 and 2006, respectively. He is currently an Associate Professor with the Università degli Studi G. d'Annunzio Chieti–Pescara, Pescara. He leaded research and development work packages is several EU and national funded projects in the area of cultural heritage and digital humanities. He is the author of more than 70 research papers published in refereed international journals and conferences. His research interests include semantic web, knowledge representation, information extraction, recommendation systems, machine learning, and deep learning.

ALESSANDRO CUCCHIARELLI received the master's degree in electronic engineering from the University of Ancona (now Università Politecnica delle Marche), Italy, in 1985. He is currently an Associate Professor with the Department of Information Engineering (DII), Università Politecnica delle Marche. He has been a member/coordinator of many research teams working on projects funded by the EU and the Italian Ministry of Research (MURST/MIUR). He is the author or co-author of about 110 papers on refereed journals, conference proceedings, and book chapters. His research interests include the application of natural language processing techniques to the automatic extraction of information from the web, to domain ontology definition, to the definition of tools and metrics for social network analysis and to cybersecurity.

LUCA SPALAZZI received the M.Eng. degree in electronic engineering, in 1989, and the Ph.D. degree in artificial intelligent systems, in 1994. He was a Consultant with IRST-FBK, Trento, Italy, from 1991 to 1993, and in 1997. He was a recipient of the Scientific National Italian Habilitation as a Full Professor, in 2020. He is currently an Associate Professor with the Università Politecnica delle Marche, Italy. His research has been supported by the European Union, the Italian Minister of University and Scientific Research, the Austrian Research Promotion Agency (FFG), the Italian ISP for research and education institutions (GARR), Regione Marche, and Provincia di Ancona. He is the author of more than 120 research papers published in refereed international journals and conferences. His research interests include both formal methods and machine learning applied to cybersecurity and distributed ledger technologies.