Journal of Network and Systems Management (2020) 28:1694–1721
https://doi.org/10.1007/s10922-020-09554-9

A Two-Stream Network Based on Capsule Networks and Sliced Recurrent Neural Networks for DGA Botnet Detection

Xinjun Pei[1] · Shengwei Tian[2] · Long Yu[3] · Huanhuan Wang[2] · Yongfang Peng[2]

Received: 4 October 2019 / Revised: 1 May 2020 / Accepted: 7 July 2020 / Published online: 20 July 2020

Abstract

With the development of Internet technology, botnets have become a major threat to most of the computers over the Internet. Most sophisticated bots use Domain Generation Algorithms (DGAs) to automatically generate a large number of pseudo-random domain names in Domain Name Service (DNS) domain fluxing, which can allow malware to communicate with Command and Control (C&C) server. To cope with this challenge, we built a novel Two-Stream network-based deep learning framework (named TS-ASRCaps) that uses multimodal information to reflect the properties of DGAs. Furthermore, we proposed an Attention Sliced Recurrent Neural Network (ATTSRNN) to automatically mine the underlying semantics. We also used a Capsule Network (CapsNet) with dynamic routing to model high-level visual information. Finally, we emphasized how the multimodal-based model outperforms other state-of-the-art models for the classification of domain names. To the best of our knowledge, this is the first work that the multimodal deep learning have been empirically investigated for DGA botnet detection.

Keywords Two-stream network · Capsule network · Sliced recurrent neural network · Attention · Domain Generation Algorithms

1 Introduction

Most of the network security configurations allow the DNS data to pass through. Attackers often embed malware commands in DNS data and manage installed malware through the C&C server. Most of these malwares like botnets use Domain Generation Algorithms (DGAs) to dynamically generate a large number of pseudorandom domain names. These domain names are called algorithmically-generated domains (AGDs), some of which are selected as the masks of malware commands

✉ Shengwei Tian
tianshengwei@163.com

Extended author information available on the last page of the article

and used to connect with C&C server. In order to completely shut down such a botnet, defenders need to intercept all AGDs generated by the malware.

Existing solutions are largely based on the linguistic features to build the models for DGA botnet detection. Unfortunately, using linguistic properties has a potential drawback because they may be bypassed by the malware authors, while deriving a new set of features is rather challenging. Some techniques incorporate contextual information (such as manual features [1, 2]) to further improve performance. However, this is a time consuming process and costly measure task, which cannot meet the needs of many real-world security applications that require real-time detection and prevention [3]. To address this challenge, we proposed a lightweight semantics and visual feature extraction, and conducted a set of experiments for confirming the validity of this feature extraction. This feature extraction is performed based on an in-depth analysis of the semantics and visual features, which is much simpler than existing manual features-based DGAs detection methods. This strategy allows a very large number of DGA domain names to be scanned quickly.

The visual features are spatially rich features that are able to encode the visual concepts of domain names without dissection the malware, when compared to other well-studied features such as traffic features and linguistic features. Furthermore, the intuition of using visual features is the inconsistent nomenclature between the normal and DGA domain names, as much of the previous work in DGA detection shown that many DGA domain names composed of random characters or words, which are different from legitimate domain names. Thus, the visualized images of normal and DGA domain names should be significantly different visually. The basic motivation is to identify the related malicious domains structure, which may be useful in identifying well-known DGAs. The introducing of visual feature helps to find the hidden spatial sequence patterns. This is almost similar to human visual systems, hence yields better accuracy than existing methods. We believe that the proposed methodology along with the results will contribute a benchmark on forthcoming DGA botnet detection proposals and research endeavors.

The challenge to catch malicious domain names that are generated dynamically has led to the recent interest in detecting DGA botnets using deep learning algorithms. In contrast to the traditional approaches, the deep learning algorithms can learn features automatically, instead of relying on manual or expert-defined features. Traditional deep learning algorithms [4], such as 1-dimensional convolution neural network (CNN [5]) and long short term memory (LSTM [6]), which are commonly used in detecting DGA botnets, can handle contextual relationships or local order information. However, by using a single model (such as CNN and LSTM) or a traditional sequential joint model (such as CNN+LSTM [4]), it is easy to lose deep features during feature mining. For example, when a traditional LSTM processes long sequence data, the front part of the features in the sequence are lost, resulting in an inability to completely capture the deep features and worsening the classification effects. Thus, this is important for researchers to find a better algorithm to learn and store the useful information of DGA domain names. To address this challenge, we proposed a novel Two-Stream network-based deep learning framework, which enforces the learning of correlation and correspondence between textual semantics and visual concepts. To highlight the significance of this study, we compared the performance of our framework with those of other existing deep learning-based methods. Overall, compared with the six baselines (see Sect. 4 for details), the performance gain of the F-score is about 0.08% to 0.82% on the four datasets.

Our goal in this paper is to develop a deep learning-based framework for detecting DGA botnets without any reverse engineering for malware. Furthermore, we applied the Two-Stream network with two independent encoders to these two tasks and measured its performance on the multimodal features from the perspective of a network administrator. In the proposed Two-Stream network, one stream is utilized to encode the semantic features and the other is applied for the visual features. Projecting the feature vectors of semantic and visual into a shared vector space through the two-stream network can provide strong associations between textual semantics and visual concepts, which has the potential to improve the ability of feature extraction. This strategy allows us to obtain good performances while keeping the complexity as low as possible.

In summary, we mainly make the following contributions:

- We introduced how to develop multimodal information in detecting DGA botnets. To the best of our knowledge, this is the first study that aims to extract multimodal information consisting of textual semantics and visual concepts.
- We proposed a novel deep learning-based framework to encode different types of multimodal information. The proposed TS-ASRCaps can automatically learn multimodal representations from the data, bypassing the human effort of feature engineering.
- From a practical perspective, the proposed method is attractive, due to its ease of implementation, acceptable computational complexity, and high execution efficiency. The experimental results show that the proposed model outperforms the state-of-the-art methods significantly.

The rest of this paper is organized as follows. The next section summarizes previous research in detecting DGA botnets. Section 3 outlines our approach, including feature extraction and classification. Section 4 presents our experimental results. Section 5 concludes the paper.

2 Related Work

One of the most common approaches for DGA botnet detection is taking the tasks as binary and multiclass classification, where each domain is assigned with a label. Many methods have focused on the analysis of DNS traffic to recognize botnets [3, 7]. These DNS traffic-based methods require the DNS traffic data from a top-level domain name server or a recursive resolution server. Schiavoni et al. [7] proposed a mechanism called Phoenix, which characterizes the DGAs using a combination of string and IP-based features. Phoenix used a set of fingerprints to label new DGA domains. Mowbray et al. [8] proposed a procedure for DGA detection, which is used to reveal and identify client IP addresses with an unusual distribution of second-level string lengths in DGA domain names (from the query data of DNS). Based on observation time and known seeds, Sivaguru et al. [9] selected data for test and training sets. In this study, they evaluated the robustness of tree ensemble models based on manual features and deep neural networks that learn features automatically from domain names. However, these methods have limited generalization capabilities in the modeling process.

In addition to the above studies, a more effective approach involves the use of deep learning techniques for identifying the DGA domains. Deep learning approaches like recurrent neural network (RNN) and convolutional neural network (CNN) have recently been proposed. These deep learning-based methods significantly outperform traditional machine learning-based methods [1, 10–12] on classification accuracy, at the price of increasing the complexity of training the model and requiring larger datasets. A typical approach is to feed the original characters of a domain name as the semantic features into deep learning model without the need for expert features [13–16] so that the model can automatically learn and classify, which also facilitates real-time detection of DGAs. In this case, all extracted character features are first converted to numbers, and then truncated or complemented to a fixed length.

Furthermore, CNN has been widely used in fields like image classification and video recognition. Recently, Catania et al. [5] provided a performance analysis and comparison of CNN designed specifically for DGA detection. Additionally, RNN has been successful in DGA botnet detection, since it can capture local information. To overcome the multiclass imbalance problem, Tran et al. [6] proposed a novel cost-sensitive LSTM, called LSTM. MI. They introduced a cost item into backpropagation learning procedure to measure the identification importance among classes. However, RNN-based methods still suffer from the defect of unable to completely record the context information in long text. Yu et al. [4] discussed two types of deep neural networks (including CNN and LSTM) based on character features for DGA domains detection. This method can be automatically identify and learn semantic knowledge.

Overall, many methods for DGA botnet detection had achieved good performance using deep learning model as a classifier. These methods are used to decide whether the given domain is a malicious domain. However, they are only conducted on the semantic features at character level without considering visual features. For DGA botnet detection, predictive accuracy should generally strongly increase when adding additional visual information.

3 Our Approach

3.1 Problem Definition

We formulated our tasks as binary and multiclass classification. Given a set of $S=\left\{\left(x^i,y^i\right)\right\}_{i=1}^{N}$, where x^i as the model input represents an instance of domain, and y^i denotes the label of the instance.

Our goal is to find a proper function $f(\cdot)$, which is the score predicting the class assignment for an instance. Finally, the output of the function with the largest probability is taken as the final class predicted by the classifier.

3.2 The Overview of TS-ASRCaps Framework

In this paper, a deep learning framework (TS-ASRCaps) of feature extraction and detection of DGAs based on multimodal information is proposed. This framework conducts two major processes; feature analysis (Sect. 3.3) and classification (Sect. 3.4). More concretely, we designed a novel ATTSRNN- and CapsNet-based Two-Stream network to model the semantic and visual knowledge. By combining natural language processing technology with image processing technology, the semantic and visual features are first extracted from domain names, and then fed them into the two streams of the Two-Stream network, respectively. Lastly, the Two-Stream network is evaluated on the fused classification score, and a classification loss is adopted for optimizing the whole network.

3.3 Feature Analysis

The accuracy of classification is directly affected by the quality of the extracted features. The multimodal vectors are divided into two types: semantic- and visual-based vectors.

3.3.1 Semantic Features

As previously noted, some deep learning methods based on character features have been proposed, which can be considered as classification of short character strings. Previous research has demonstrated the effectiveness of character features. We extracted DGA domain name characters as semantic features, which was inspired by the reports of previous work [4, 6]. Thus, each character contained in a domain name is represented by a number.

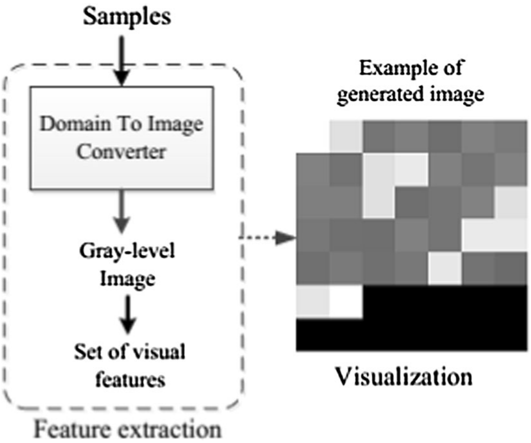
3.3.2 Visual Features

Additionally, we extracted the visual features from the domain names. A typical approach [16] is represented executable malware binary files as one-channel gray-scale images via scanning and converting every bit value in binary files into an image pixel. For example, Su et al. [17] proposed a light-weight approach to classify malware in IoT environments. After generating images, they firstly analyzed these converted binary images and then applied a lightweight convolutional neural network for detecting DDoS malware.

Inspired by this, we extracted the gray-scale images from domain names and classified them according to the similarity of image texture. This strategy allows researchers to intuitively understand the spatial patterns and structures of domain names. So this visual feature will provide an effective input for subsequent model learning. Our main assumption is that image samples from different DGA families have distinct texture characteristics. Because the different DGA domains are generated by different algorithms. Furthermore, we treated the domain names as binary data [17–19]. Each character is represented by the corresponding one-channel gray-scale image pixel, and each transformed gray-scale image contains some layouts and textures. Figure 1 illustrates the visualization process of extracting visual features. "Domain to Image Converter" is first transformed a domain into a gray-scale image, and visual features are then represented and stored in a database. Specifically, converting domains to the corresponding images only requires creating the input vectors to the subsequent model learning, which is a very fast operation.

Algorithm 1 illustrates the visualization process for converting the DGA domain names to gray-scale images. Firstly, the characters of domain name are stored in a 1-D array. We set the length of array L = 50. Furthermore, domain names longer than 50 characters would get truncated from the 50-th character, and any domain names shorter than 50 characters would get padded with the special character '0′ till their lengths reached 50. This array can be treated as a 2-D matrix of a specified width and height. For simplicity, the width and the height of the image are fixed (Algorithm 1, Line 2–3). Finally, we converted the characters of domain name to the pixels of image (Algorithm 1, Line 4–10).

Fig. 1 Example of the visualization process



Algorithm 1. Visual features extraction

```
Input: DGA domain names
Output: Gray-scale Images
function grayscale ()
    Rows ← get_Grayscale_Width ()
    Columns ← Grayscale_size/Rows
    init array[Rows][Columns]
    for i=0 → Rows do
        for j=0 → Columns do
            array[i][j] ← convert characters to pixels
        end for
    end for
    GrayscaleImages ← convert array[Rows][Columns] to gray-scale images
    return GrayscaleImages
end function
```

3.4 Two-Stream Architecture

In this part, we introduced the Two-Stream architecture that includes three components, namely, Attention (ATT), Sliced Recurrent Neural Network (SRNN) and Capsule Network (CapsNet). The overall flow of the proposed TS-ASRCaps is shown in Fig. 2. Specifically, the choice of the optimized Two-Stream architecture is based on the experiments in Sect. 4.

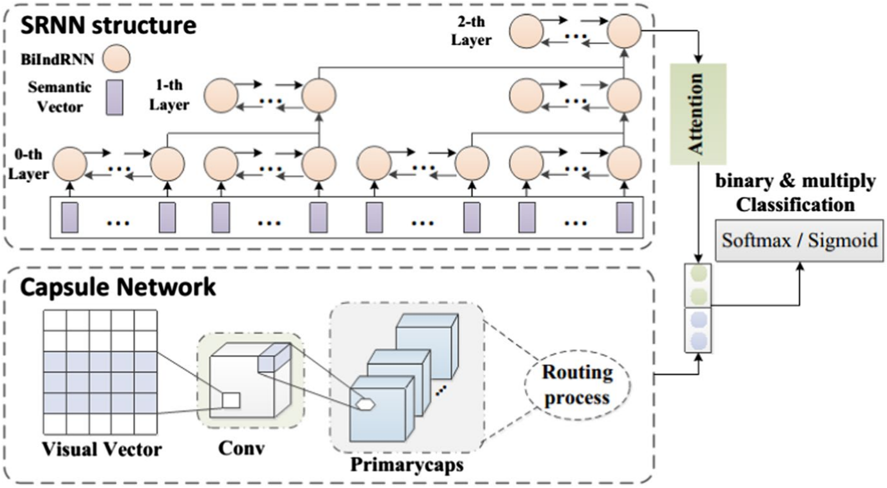


Fig. 2 Flow diagram of the TS-ASRCaps

3.5 The Semantic Stream

3.5.1 Embedding Layer

In order to improve the statistical quality of the model, one-hot is a typical method for handling categorical data, which encodes words and stores them into a sparse matrix that is used as the input of a deep learning model. However, for the one-hot encoding method, a serious problem is the high dimensionality of the data and the curse of dimensionality. Instead of the one-hot encoding method, we encoded semantic features into dense vectors of real values by applying a trainable embedding layer, which is helpful for dealing with high dimensionality and data sparsity. In such a case, the generated embedding maps the semantic-knowledge into a fixed $n \times m$ matrix, where n represents the number of the semantic features. Each semantic feature is stored in a $1 \times m$ vector represented by a row in the matrix.

3.5.2 Independently Recurrent Unit

In order to prevent the gradient exploding and vanishing in recurrent neural network (RNN), the variant of RNN, referred to as independently recurrent neural network (IndRNN), have been proposed [20], which allows the network to learn long-term dependencies. Furthermore, the LSTM was developed to address the gradient explosion and disappearance problems effectively when the network converges, but the use of hyperbolic tangent and the sigmoid action functions results in gradient decay over layers [20]. Compared with the LSTM, the neurons in the same layer of IndRNN are independent of each other and connected across layers, and the gradient can be effectively propagated at different time steps. Through the gradient reverse propagation of time, the IndRNN makes network memory more effective in learning content and processing long sequences. Thus, the IndRNN can make full use of the long-range information while preventing the gradient explosion and disappearance problems. Furthermore, the gradient can be effectively propagated at different time steps, resulting in faster processing compared to LSTM.

This standard IndRNN structure with a non-saturated function σ such as relu as activation function can be described as:

$$\mathbf{h}_t = \sigma\left(\mathbf{W}x_t + \mathbf{U} \odot \mathbf{h}_{t-1} + \mathbf{b}\right), \tag{1}$$

where $\odot$ represents the Hadamard product, $\mathbf{U}$ and $\mathbf{W}$ are the recurrent weight and the input weight, respectively. Since spatial patterns are aggregated independently (i.e. through $\mathbf{W}$) over time (i.e. through $\mathbf{U}$), neurons at different times in the same layer are independent of each other.

However, the IndRNN only uses forward context information during the processing sequence. To improve the modeling ability to integrate forecast knowledge, the improved bidirectional independently recurrent neural network (Bi-IndRNN) is used in this paper, which can capture both forward and future context information. The standard Bi-IndRNN structure is shown in Fig. 3. It can be described as:

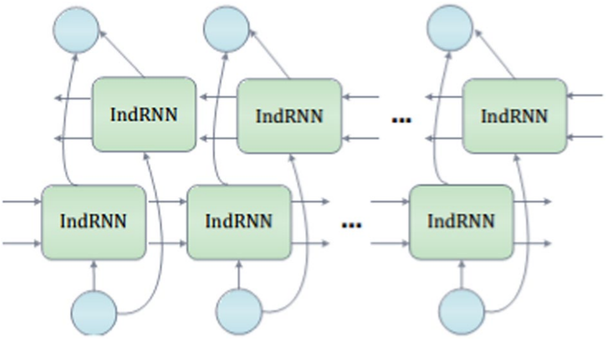


Fig. 3 The BiIndRNN structure

$$\vec{\mathbf{h}}_t = \sigma\left(\mathbf{W}x_t + \mathbf{U} \odot \vec{\mathbf{h}}_{t-1} + \mathbf{b}\right), \tag{2}$$

$$\overleftarrow{\mathrm{h}}_t = \sigma\left(\mathbf{W}x_t + \mathbf{U} \odot \overleftarrow{\mathbf{h}}_{t-1} + \mathbf{b}\right), \tag{3}$$

where each input for the forward-to-future and future-to-forward direction are associated with a hidden state $\vec{\mathbf{h}}_t$ and a hidden state $\overleftarrow{\mathbf{h}}_t$ corresponding to $\vec{\mathbf{h}}_t$. Next, we concatenate the two vectors to form the output of the BiIndRNN. In doing so, each hidden state $\mathbf{h}_t$ contains information about the whole input sequence.

3.5.3 Sliced Recurrent Neural Networks

In order to improve the traditional RNN connection structure, a new RNN structure called sliced recurrent neural networks (SRNNs) has been proposed [21, 22], which have the ability to obtain high-level semantic information of the input sequences, not just the character-level information.

The input sequence is sliced into several minimum subsequences with equal length. At each layer, the SRNN can work on each subsequence simultaneously through the improved RNN connection structure. Furthermore, the Bi-IndRNN is integrated as the recurrent unit of SRNN and is used for each subsequence, so that Bi-IndRNN can be computed in parallel, which brought the superiority of both Bi-IndRNN and SRNN into full play. The hidden state of each minimum subsequence on the 0-th layer can be described as:

$$\mathbf{h}_t^1 = BiIndRNN^0\left(mss^0_{t-l_0\sim t}\right), \tag{4}$$

where *mss* denotes minimum subsequences on the 0-th layer, *t* denotes the length of each subsequence, l_0 denotes the minimum subsequence length, and the Bi-IndRNN is used on each layer. On the *p-th* layer, the last hidden state of the subsequences can be described as:

$$\mathbf{h}_t^{p+1} = BiIndRNN^p\left(h_{t-l_p}^p \sim h_t^p\right), \tag{5}$$

where l_p denotes the subsequence length of the *p-th* layer.

In this way, information can be obtained in many short subsequences and important information is then transmitted in parallel through the multiple-layers structure from the *0*-th layer to the top layer.

3.5.4 Attention Layer

For knowledge distillation, we stacked an attention module [23] on the top layer of the SRNN, which pays more attention to the quality of the semantic features of the SRNN hidden-layer. The attention mechanism is used to extract such characters that are important to the meaning of the semantic expression since the contribution of each character to the semantic expression of a domain name is different. Thus, some critical semantic features are obtained by considering the probability weight distribution. Furthermore, the representations of those informative characters are then aggregated into vectors. The context vector $\mathbf{s}_i$ for the attention allocation coefficient a_{ij} is generated as follows:

$$\mathbf{s}_i = \sum_{j=1}^{n} \alpha_{ij} ann_j. \tag{6}$$

The weight coefficient of attention mechanism is calculated as follows:

$$\alpha_{ij} = \exp\left(e_{ij}\right) \Big/ \sum_{k=1}^{n} \exp\left(e_{ik}\right), \tag{7}$$

where

$$e_{ij}=a\left(\mathbf{h}_{i-1}, ann_j\right)$$

is an alignment model which scores how well the inputs around position j and the output at position i match [23]. The score is based on the hidden state h_{i-1} of the previous layer and the j-th annotation ann_j.

3.5.5 The Visual Stream

The visual stream study also begins with an embedding layer. All of these visual features are treated as the specially learnt words, and then embed them into a specific visual vector space. This strategy enables us to better identify the combinations and patterns of features.

The capsule network [24–26] is effective for image-related inference tasks (e.g., image classification, etc.). We introduced a version of CapsNet to extract visual features from gray-scale images that help improve the classification accuracy. Our main assumption is that the CapsNet will able to successfully detect DGA botnets using raw pixel values extracted from domain names. Each capsule performs complex internal calculations on sample inputs and learns how to represent and reconstruct a given sample like an autoencoder. Compared with CNN [5], the CapsNet converts scalars into vectors, which can better store features. Furthermore, the dynamic routing algorithm is used to ensure a more accurate output of low-level capsule vectors to higher-level parent capsules. By doing this, the dynamic routing-based capsule network can encode the intrinsic spatial relationships between a part and a whole knowledge. The above advantages show that CapsNet is a promising architecture against to standard CNN.

3.5.6 Convolutional Layer

In this network, a standard convolutional layer is used to generate effective features from different positions of the embedded vector matrix in the previous embedding layer. The convolutional operations are computed by:

$$z_l = f\left(\mathbf{W}^a \circ X_{l:l+k-1} + b\right), \tag{8}$$

$$\mathbf{Z}^{(i)} = \left[z_1, z_2, \dots, z_{L-k+1}\right], \tag{9}$$

where ∘ denotes element-wise multiplication. The convolutional filter is denoted by $\mathbf{W}^a \in R^{K_1 \times V_1}$, where K_1 and V_1 are used to describe the window size of the convolutional filter. All extracted features are collected into one feature map by sliding the filter over the embedded vector matrix.

3.5.7 Capsule Layer

Next, a convolutional capsule layer with a group-convolution operation is used to transform the feature maps into primary capsules. Capsules apply a subset of the filters used in the previous layer to represent each element in the current layer. More formally, features at same position from all feature maps are encapsulated into a corresponding capsule by the 1 × 1 filters $\mathbf{W}^b = \left\{w_1, \dots, w_v\right\} \in R^{V_2}$ shard across different windows. Thus, a capsule vector $\boldsymbol{p}_i$ is computed by:

$$p_{ij} = z_i \cdot w_j \in R, \tag{10}$$

$$\boldsymbol{p}_i = g(\left[p_{i1}, p_{i2}, \cdots, p_{iV_2}\right]) \in R^{V_2}, \tag{11}$$

where “[]” is the concatenation operator, V_2 is the dimension of a capsule vector, and g is a non-linear squashing function. Specially, the length $\left\|p_i\right\|$ of each capsule is constrained to the unit interval [0, 1] by the squashing function:

$$g(x) = squash(x) = \frac{\|x\|^{\mathbf{2}}}{\mathbf{1}+\|x\|^{\mathbf{2}}} \frac{x}{\|x\|}. \tag{12}$$

By a linear transformation matrix $\mathbf{W}^c$, the prediction vector $\hat{\boldsymbol{u}}_{j|i}$ from its *i-th* child capsule in the first capsule layer to the *j-th* parent capsule in the subsequent capsule layer is generated by:

$$\hat{\boldsymbol{u}}_{j|i}=\mathbf{W}^{c}\boldsymbol{u}_{i}. \tag{13}$$

Then, the high-level capsule $\boldsymbol{v}_j$ is calculated as a weighted sum over all prediction vectors $\hat{\boldsymbol{u}}_{j|i}$. The capsule

$$\boldsymbol{v}_j=g\left(\sum_{i} c_{j|i}\hat{\boldsymbol{u}}_{j|i}\right), \tag{14}$$

where c_i is the coupling coefficients that are determined by the dynamic routing process.

3.5.8 Fusion Vector

The outputs of the two streams are directly combined together by a concatenation operation "[]", so that the semantic and visual features are closely interacted. Furthermore, the multimodal vectors $\boldsymbol{m}$ can be obtained by:

$$\boldsymbol{m}=[\boldsymbol{s},\boldsymbol{v}]. \tag{15}$$

After fusion, it is passed on to further layers. This multimodal vectors are used to calculate the class probabilities for classification task.

Furthermore, the multimodal vectors m can be invoked as features for the binary or multiclass classification. We let the Sigmoid/Softmax function operate on the multimodal vectors.

For the binary classification, the Sigmoid function is used as the output layer to model the binary probabilities.

$$y_{binary} = Sigmoid\,(\boldsymbol{m}). \tag{16}$$

For the multiclass classification, the Softmax function is used as the output layer to model the multiclass probabilities:

$$y_{multiclass} = Softmax\,(\boldsymbol{m}). \tag{17}$$

Our network is trained by minimizing the cross entropy loss, and the objective function is optimized using the gradient-based optimization algorithm Adam. Having the same architecture in both the binary and multiclass classification subtasks makes the development and the evaluation of a given design simpler.

4 Experiments and Evaluation

For a comprehensive assessment, we evaluated the TS-ASRCaps in a binary experiment (Whether DGA?) and multiclass experiment (Which DGA?) by describing the details of our experimental setup and evaluation metrics. We conducted two evaluation studies to answer the following research questions:

- *For Study I* How accurate is TS-ASRCaps in detecting DGA and non-DGA, and how does it compare to other state-of-the-art peer approaches that address the same problem?
- *For Study II* Does TS-ASRCaps have the ability to distinguish one DGA algorithm from another with high precision?

4.1 Dataset and Metrics

Dataset Our model is evaluated on three widely used benchmark datasets containing both DGA and normal domains. The normal domains were obtained from the Alexa top 1 million domains [27]. The DGA domains were obtained from the repositories of DGA domains of OSINT [28], Lab360 [29], and Andrey Abakumov [30]. Table 1 lists the descriptions of the three datasets, where #Num is the number of DGA and normal domain names. The OSINT DGA feed (used in our experiment) from Bambenek Consulting consists of thirty families of DGAs with a varying number of examples from each class, which was downloaded on the following dates: April 11, May 7, and July 10, 2019.

Along with these three public datasets, we also collected a lot of domains generated by real DNS traces in Jan 2020 from our university. It is referred as XJU dataset. For all the users connected to the university network, their DNS queries are sent to the DNS server. We collected all the DNS traffic by mirroring the ports of the DNS server. Specially, after deduplication, only 88,913 of the queried domains (including 86,075 normal domains and 2838 DGA domains) are unique. Some statistics for this dataset are listed in Table 1.

In our experiment, 80% of the samples being randomly selected as the training set, and the remaining as the test set. Note that, in order to avoid causing biases or overfitting, this evaluation study did not involve any re-sampling. These two sets do not have any common samples.

Evaluation metrics For evaluating the classification results, we used some common performance evaluation metrics to quantify numerically the performance of the classifier, such as accuracy (ACC), precision (P), recall (R), F-score (F). The goal of any DGA classification research is to achieve a high value for F-score (F). Additionally, following the work of [6], we use the micro and macro average to averaging results over classes. For micro average, smaller classes are considered less than larger classes in the average, which is a better performance predictor for this paper.

Table 1 Main datasets used in our evaluation studies

Datasets	Non-DGA		DGA	
	Source	#Num	Source	#Num
OSINT	Alexa	1000,000	OSINT	956,241
Lab360	Alexa	1000,000	Lab360	1174,481
AR	Alexa	1000,000	Abakumov	694,173
XJU	University	86,075	University	2838

For macro average, all classes are averaged regardless of the number of elements in each individual class.

Experimental Set-up Additionally, we also provided the parameter settings of TS-ASRCaps as shown in Table 2.

Among them, the Semantic and Visual denote the dimension of the semantic and visual features, respectively. The Embedding denotes the dimension of the generated embedding vectors. The SRNN-layers, Bi-IndRNN-layers, Attention-layers and CapsNet-layers indicate the number of SRNN, Bi-IndRNN, Attention and CapsNet layers, respectively. The Bi-IndRNN is used as the recurrent unit of SRNN. The Bi-IndRNN-units indicates the number of Bi-IndRNN neurons. The Capsule-numbers indicates the number of capsules used in the capsule layer. The Capsule-dimensions denotes the dimension of a capsule. The Dropout indicates that the dropout technique is used to overcome over-fitting during training. The Batch indicates the batch amount. The Epochs indicates the number of iterations for the model training. In addition, the loss function used in TS-ASRCaps is the cross-entropy. The optimizer used in TS-ASRCaps is the Adam and the learning rate of the Adam is 0.001.

4.2 Study I: DGA Botnet Detection

The Performance Comparison for DGA Botnet Detection For comparison, we start from the original proposals as can be found in the literature [31], which provides the five state-of-the-art deep learning models for DGA botnet detection, including Endgame, Invincea, CMU, MIT, and NYU. We reimplemented these models as a set of baselines. In addition, following the work of [6], we also reimplemented the LSTM-MI architecture. It is also worth mentioning that Endgame, CMU, MIT, and LSTM-MI are also used in the recent research [9]. Specially, all

Table 2 Hyper parameter setting

Parameters	Values
Semantic	64
Visual	64
Embedding	100
SRNN-layers	3
SRNN-recurrent-units	Bi-IndRNN
Bi-IndRNN-layers	1
Bi-IndRNN-units	64
Attention-layers	1
CapsNet-layers	1
Capsule-numbers	4
Capsule-dimensions	32
Dropout	50%
Epoch	20
Batch	128
loss function	Cross-entropy
optimization algorithm	Adam

baselines were built using the descriptions and specified parameters from existing papers [6, 31]. Overall, we compared our proposed model against the following six baselines:

- Endgame Model [4] [31]: A long short-term memory network (LSTM) was designed specifically for DGA detection.
- CMU Model [31]: A standard bidirectional language model for DGA detection consists of a forward and backward long short-term memory network.
- NYU Model [4, 5, 31]: A standard convolutional neural network (CNN) was originally proposed by [32], and adapted for DGA detection by [4, 5], and [31].
- Invincea Model [31]: An extended model of CNN with parallel architecture was adapted by [31].
- MIT Model [31]: A hybrid neural network consists of an embedding layer, a CNN layer, and an LSTM layer, which was originally proposed for character-level text processing by [33].
- LSTM-MI [31]: The cost-sensitive LSTM model for DGA detection was proposed by [6], which introduces the cost items into the backpropagation learning procedure to take into account the identification importance among classes.

These models are used independently to model multimodal information without using the Two-Stream mode. The results are listed in Table 3. All models performed well on the four benchmark datasets. We can see that the LSTM-MI, Endgame, and NYU are closest to our TS-ASRCaps in overall performance, which fully demonstrates the powerful generalization ability of these models.

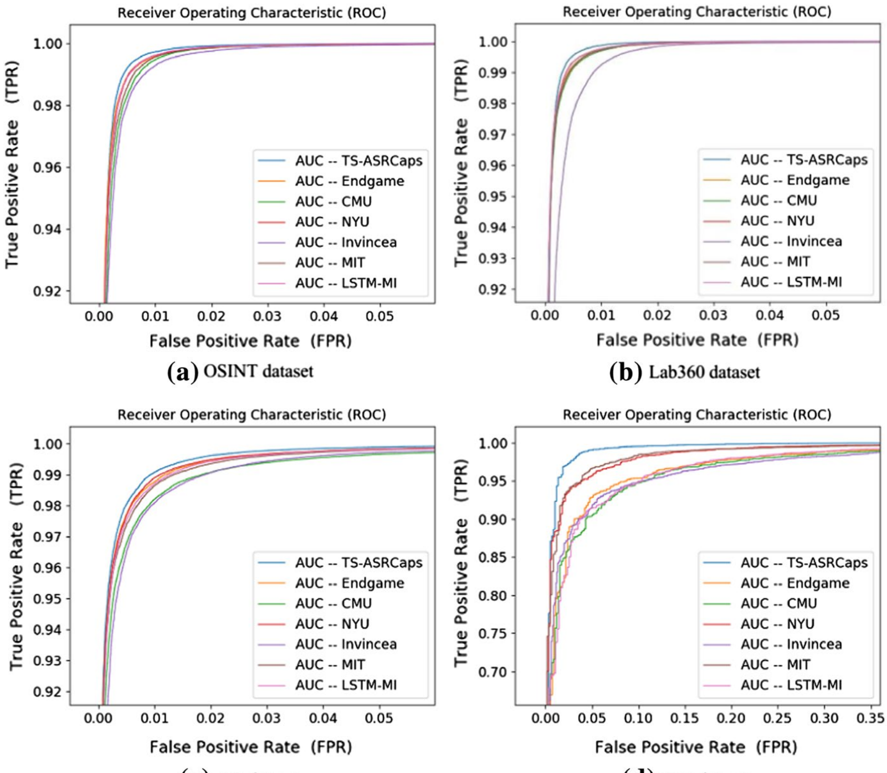
Table 3 TS-ASRCaps versus baselines for DGA detection on the four benchmark datasets (the size of dataset used in this experiment is described in Table 1)

Dataset		Endgame	CMU	NYU	Invincea	MIT	LSTM-MI	TS-ASRCaps
OSINT	Acc	99.28	99.2	99.32	99.14	99.27	99.32	99.4
	P	99.35	98.79	99.22	99.02	99.06	99.21	99.26
	R	99.25	99.65	99.46	99.3	99.52	99.47	99.58
	F	99.3	99.22	99.34	99.16	99.29	99.34	99.42
Lab360	Acc	99.36	99.36	99.42	99.02	9.37	99.4	99.51
	P	98.97	99.12	99.2	98.22	98.99	99.43	99.38
	R	99.66	99.49	99.54	99.68	99.64	99.26	99.57
	F	99.31	99.31	99.37	98.94	99.31	99.35	99.47
AR	Acc	98.88	98.66	98.95	98.63	98.85	98.89	99.05
	P	99.25	98.87	99.23	98.77	98.95	99.06	99.14
	R	98.85	98.87	99	98.91	99.09	99.06	99.26
	F	99.05	98.87	99.11	98.84	99.02	99.06	99.2
XJU	Acc	97.96	97.62	98.31	97.67	98.59	97.89	99.24
	P	98.75	97.8	98.51	98.68	99.28	99.14	99.61
	R	99.15	99.77	99.76	98.91	99.26	98.67	99.6
	F	98.95	98.78	99.13	98.8	99.27	98.91	99.6

Although their performances better than the other methods, but they are not yet comparable to the results given by the TS-ASRCaps. The TS-ASRCaps substantially outperforms all baselines, since it takes into account the mutual relationship between semantic and visual. Figure 4 shows the ROC curves for the four benchmark datasets. This shows that the TS-ASRCaps has superior performance in detecting DGA botnets.

Is the Two-stream network redundant*?* In this part, we conduct experiments to evaluate: (1) the effectiveness of multimodal. (2) the contributions of each component of TS-SRACaps. As TS-SRACaps comprises a set of contiguous components, such as Attention Mechanism (ATT), Sliced Recurrent Neural Network (SRNN) and Capsule Network (CapsNet), we designed four models to investigate the necessity and benefits of these components.

- OS-CapsNet (One-Stream CapsNet): OS-Caps is designed to test whether visual features and CapsNet for DGA detection are necessary, which adopt one branch of the CapsNet to learn the visual information.



(a) OSINT dataset

(b) Lab360 dataset

(c) AR dataset

(d) XJU dataset

Fig. 4 ROC curves for the four benchmark datasets

- OS-CNN (One-Stream CNN): To verify the necessity of CapsNet, we also designed a baseline OS-CNN to learn the visual information.
- OS-SRNN (One-Stream SRNN without Attention Mechanism): To verify the necessity of semantic features and SRNN, we designed a baseline OS-SRNN to learn the semantic information, which uses one branch of the SRNN.
- OS-ATTSRNN (One-Stream SRNN with Attention Mechanism): Unlike the OS-SRNN model, one branch of SRNN with an attention mechanism (OS-ATTSRNN) is used to learn semantic information and to model the hidden contextual by calculating the attention weights, which verifies the necessity of the attention mechanism. OS-ATTSRNN is a straightforward combination of the SRNN and attention models.

As shown in Table 4, the OS-CapsNet performs the best in this One-Stream mode. This is the only configuration that uses a CapsNet as it is not tested in combination with the SRNN. Meanwhile, the OS-CapsNet in the OSINT dataset has an accuracy of 99.26%, which is a significant improvement over the OS-CNN. This is because the OS-CapsNet can encode the intrinsic spatial relationships between a part and a whole knowledge. Specially, an ensemble of SRNN with attention (i.e. OS-ATTSRNN) substantially improves the overall performance compared to the OS-SRNN, with an accuracy of 98.69% and an F-score of 99.32% on the XJU dataset. This is because the OS-ATTSRNN can capture the importance of each context character while the OS-SRNN cannot. The result listed in Table 4 affirmed the effectiveness of the proposed semantic and visual features.

As we expect, the TS-ASRCaps outperformed all baselines since the ensemble method benefits from the incorporation of the Two-Stream network outputs. The

Table 4 Ablation studies of TS-ASRCaps on the four benchmark datasets

Dataset		OS- CNN	OS-CapsNet	OS-SRNN	OS-ATTSRNN	TS-ASRCaps
OSINT	Acc	99.13	99.26	98	98.03	99.4
	P	98.95	99.26	97.51	97.64	99.26
	R	99.35	99.29	98.6	98.52	99.58
	F	99.15	99.27	98.05	98.08	99.42
Lab360	Acc	99.11	99.44	98.34	98.42	99.51
	P	98.94	99.21	97.48	97.65	99.38
	R	99.13	99.58	98.93	98.96	99.57
	F	99.03	99.39	98.2	98.3	99.47
AR	Acc	98.11	98.84	97	96.91	99.05
	P	98.35	98.89	97.74	97.5	99.14
	R	98.44	99.15	97.17	97.26	99.26
	F	98.4	99.02	97.45	97.38	99.2
XJU	Acc	98.36	98.72	98.5	98.69	99.24
	P	98.4	98.76	99.52	99.38	99.61
	R	99.93	99.93	98.93	99.26	99.6
	F	99.15	99.34	99.22	99.32	99.6

knowledge of multimodal for the detection task can be learned from the DGA data since the semantic and visual embeddings are shared cross the network. Again, the ensembles lead to a dramatic increase in performance, showing that the ATTSRNN and CapsNet are complementary.

The Performance Comparison for the Proportion of Training and Testing Sets To demonstrate the reliability of the TS-ASRCaps, we investigated the impact of the number of training samples on classification accuracy. The experiment settings are along the same lines as previously mentioned in Sect. 4.1. We evaluated our model by training it on a small number of samples while testing on unseen samples.

Note that there are no common sample between the two sets. In this experiment, we found that the number of training samples plays a crucial role in the performance of the model. As shown in Fig. 5, we changed the proportion of training and testing sets. We can see that, when the TS-ASRCaps used 20% of the data for training, the classification accuracy rate is 99.14%. The results indicated that the TS-ASRCaps can achieve good results in the case of a small number of training samples, demonstrating that the feature collection ability of the TS-ASRCaps is very strong. The downside is that the classification accuracy rate has not reached the best level.

Furthermore, when the number of training samples increases, the model fits the data distribution better. The very strong ability in express information will make the value from the function close to the desired target, and thus improve the performance of the model. The experimental results indicate that the TS-ASRCaps is more sensitive to the number of training samples. A modest increase in the training sample size would improve the performance of the model.

4.3 Study II: Multiclass

The Performance Comparison for Familial Classification For assessing the performances of the proposed model, we reported the familial classification results of the TS-ASRCaps against the rival methods (the hyper-parameters of all models are fixed without tuning). The precision, recall, and F-score are displayed in Tables 5

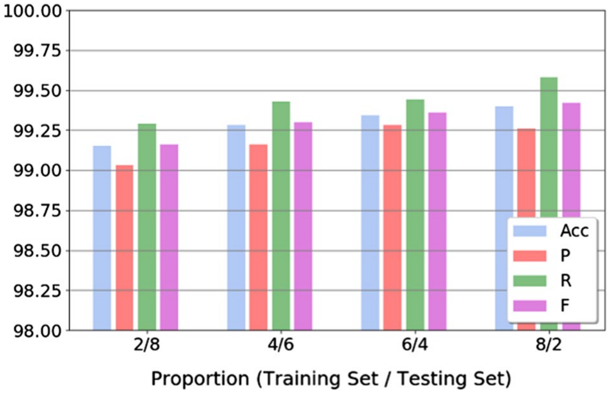


Fig. 5 The comparative classification performance in various proportions of training and testing sets

Table 5 Multiclass classification results in terms of precision, recall and F-score—part I

Family	#Num	Endgame			CMU			NYU			TS-ASRCaps		
		P	R	F	P	R	F	P	R	F	P	R	F
Aleax	1000,000	99.52	99.56	99.54	99.49	99.69	99.59	99.31	99.77	99.54	99.45	99.84	99.64
PT Goz	66,000	99.79	99.98	99.88	99.98	99.97	99.97	99.96	99.97	99.96	100	99.99	100
Necurs	49,152	93.74	84.19	88.71	94.78	83.52	88.8	94.14	84.27	88.94	97.33	81.71	88.84
Qakbot	40,000	79.03	79.98	79.5	75.98	83.2	79.43	75.38	81.92	78.52	77.41	81.19	79.26
Murofet	26,500	85.74	82.24	83.95	82.26	88.07	85.06	80.54	82.97	81.74	83.37	85.18	84.26
Tinba	66,688	93.33	99.7	96.41	94.92	99	96.92	93.62	98.89	96.18	94.33	99.26	96.73
Ramnit	56,174	78.94	91.89	84.92	81.38	90.69	85.79	79.64	90.87	84.88	81.6	90.8	85.96
Ranbyus	13,640	79.59	87.41	83.32	82.4	85.45	83.9	82.45	84.56	83.49	82.09	85.74	83.88
Dyre	7998	99.94	99.94	99.94	100	100	100	99.92	100	99.96	99.94	99.75	99.84
Hesperbot	192	0	0	0	0	0	0	0	0	0	0	0	0
Locky	5352	84.43	32.07	46.48	82.73	30.03	44.06	89.19	29.62	44.47	74.57	33.56	46.29
Wiki 25	5000	40.16	51.1	44.97	54.76	29.53	38.36	62.7	25.93	36.69	49.46	40.28	44.4
Bebloh	12,521	98.53	91.4	94.83	98.72	91.42	94.93	99.23	90.76	94.81	99.06	91.55	95.16
Kraken	8988	97.66	76.1	85.54	91.71	81.65	86.39	98.58	76.85	86.37	87.71	85.96	86.83
Chinad	1536	99.05	97.81	98.43	98.89	97.45	98.16	96.31	96.52	96.42	99.29	94.26	96.71
Pushdo	1680	91.19	86.31	88.69	94.06	87.42	90.62	91.92	80.29	85.71	96.25	87.5	91.67
Cryptolocker	1000	0	0	0	0	0	0	0	0	0	0	0	0
Vawtrak	3150	77.89	69.05	73.21	89.94	84.92	87.35	84.95	53.74	65.83	80.97	77.83	79.37
Sphinx	768	0	0	0	0	0	0	0	0	0	0	0	0
Proslikefan	600	83.33	20.49	32.89	88.64	29.1	43.82	91.43	17.68	29.63	100	18.85	31.72
Geodo	576	44.44	7.08	12.21	50	94.59	65.42	100	0.52	1.04	47.45	98.94	64.14
Padcrypt	576	100	93.46	96.62	95.9	99.15	97.5	96.34	86.81	91.33	92.54	98.41	95.38
Ramdo	2000	99.76	99.76	99.76	100	100	100	98.56	98.71	98.63	96.95	100	98.45

Table 5 (continued)

Family	#Num	Endgame			CMU			NYU			TS-ASRCaps		
		P	R	F	P	R	F	P	R	F	P	R	F
Corebot	240	100	100	100	100	98.08	99.03	98.78	100	99.39	100	97.83	98.9
Volatile	996	99.5	97.09	98.28	98.51	100	99.25	99.01	100	99.5	95.54	99.48	97.47
Bamital	240	100	100	100	100	100	100	100	98.61	99.3	100	100	100
Beebone	210	100	100	100	100	100	100	100	98.44	99.21	97.44	100	98.7
Dircrypt	720	0	0	0	0	0	0	0	0	0	0	0	0
Tempedreve	249	0	0	0	100	1.82	3.57	0	0	0	0	0	0
Fobber	600	38.81	19.85	26.26	37.04	7.81	12.9	66.67	1.16	2.27	44.44	3.15	5.88
vidro	200	0	0	0	0	0	0	0	0	0	0	0	0
Micro	–	96.59	96.59	96.59	96.8	96.8	96.8	96.55	96.55	96.55	96.82	96.82	96.82
Macro	–	69.82	63.43	64.98	73.94	66.53	67.12	73.5	60.61	62.7	70.23	66.16	66.11

and 6 for the Endgame, Invincea, CMU, MIT, NYU, and LSTM-MI. Based on the additional information provided by detecting the specific DGA malware family, anti-malware providers can validate the results, thereby making optimal detection decision with higher confidence. Our goal is to retain the high F-score on the non-DGA (Alexa) class while increasing the micro and macro averaging F-score on the DGA classes. As we can see, the TS-ASRCaps has a more balanced F-score in all classes, which explains why an ensemble of SRNN with CapsNet works, as they are complementary to one another. Figure 6 shows the multiclass classification performance of the TS-ASRCaps for each DGA family.

We discovered that some families were misclassified most of the time, such as cryptolocker, tempedreve, hesperbot, fobber, and dircrypt. These observations are along the same lines as [6]. All baselines are well recognized for some DGA families and do not perform well in other families. This is perhaps because the serious imbalance class distribution towards these families makes the classifier less likely to learn useful information. According to [6], another possible reason may be due to the uniform distribution of letters generated by those malwares. Overall, although the different families are classified, the TS-ASRCaps is still comparable to these state-of-the-art techniques. The TS-ASRCaps also has the ability to retain high F-score on the non-DGA (Alexa) class.

In addition, to analyze the weighting process of attention, we drew the attention distributions at attention layer as shown in Fig. 7. Color proceeding a domain name denotes to the weight of the attention matrix, and does not necessarily denote non-DGAs or DGAs. The attention layer acted somewhat as an optimized feature extractor on the sequences of semantic feature vectors produced from previous SRNN layer, and the cell of attention provided an indication of what the semantic feature was weighting. We can observed that after attention learning, some characters of a domain can obtain a high weight at attention layer, while others cannot. In other word, the key semantic features would be collected and irrelevant parts would be ignored. This shows our application of attention layer for the DGAs classification is effective at guiding the attention layer to select the vital semantic features.

4.4 Efficiency

Analyzing a domain name is mainly divided into two phases: one is the feature extraction phase, and the other is the classification phase. For the feature extraction phase, previous studies, such as [24], have demonstrated the effectiveness of character features for deep learning model in DGA botnet detection. These methods are more efficient than ours because only character features are extracted. Compared to them, we additionally extracted the visual features as the input of classifier. In fact, the extraction of the multimodal features is a very fast operation, and the time taken to process one sample for feature extraction is almost negligible. Another topic of discussion is the efficiency of our method in classifying DGAs. For potential users, the classification time is critical. For assessing the overall runtime, we focused on evaluating the processing time for feature extraction and then giving an average runtime for processing one sample. Furthermore, we time predation for 100 k domains,

Table 6 Multiclass classification results in terms of precision, recall and F-score—part II

Family	#Num	Invincea			MIT			LSTM-MI			TS-ASRCaps		
		P	R	F	P	R	F	P	R	F	P	R	F
Aleax	1000,000	99	99.78	99.38	99.44	99.75	99.6	99.48	99.69	99.58	99.45	99.84	99.64
PT Goz	66,000	99.88	99.9	99.89	100	99.98	99.99	99.92	99.94	99.93	100	99.99	100
Necurs	49,152	96.09	73.03	82.99	96.04	82.82	88.94	95.13	83.35	88.85	97.33	81.71	88.84
Qakbot	40,000	69.47	78.42	73.67	76.46	82.21	79.23	78.34	79.74	79.03	77.41	81.19	79.26
Murofet	26,500	77.88	58.31	66.69	80.24	87.9	83.89	80.4	88.73	84.36	83.37	85.18	84.26
Tinba	66,688	75.54	94.98	84.15	94.05	99.3	96.6	94.36	98.97	96.61	94.33	99.26	96.73
Ramnit	56,174	72.75	80.28	76.33	81.89	89.23	85.4	82.43	89.98	86.04	81.6	90.8	85.96
Ranbyus	13,640	59.56	61.41	60.47	83.04	82.53	82.78	81.73	86.27	83.94	82.09	85.74	83.88
Dyre	7998	99.87	100	99.94	100	100	100	99.92	100	99.96	99.94	99.75	99.84
Hesperbot	192	0	0	0	0	0	0	0	0	0	0	0	0
Locky	5352	95.3	21.36	34.9	76.95	36	49.05	77.85	35.22	48.5	74.57	33.56	46.29
Bedep	5000	70.7	15.61	25.57	63.27	29.39	40.13	48.6	41.2	44.6	49.46	40.28	44.4
Bebloh	12,521	94.16	88.22	91.09	99.11	91.88	95.36	98.93	90.78	94.68	99.06	91.55	95.16
Kraken	8988	97.28	76.77	85.82	88.76	85.33	87.01	91.75	81.72	86.45	87.71	85.96	86.83
Chinad	1536	96.16	69.57	80.72	98.62	98.28	98.45	98.73	97.91	98.32	99.29	94.26	96.71
Pushdo	1680	89.18	81.68	85.27	93.45	85.79	89.46	86.17	92.93	89.42	96.25	87.5	91.67
Cryptolocker	1000	0	0	0	0	0	0	0	0	0	0	0	0
Vawtrak	3150	96.36	34.81	51.15	89.55	85.58	87.52	88.89	90.68	89.77	80.97	77.83	79.37
Sphinx	768	0	0	0	0	0	0	0	0	0	0	0	0
Proslikefan	600	0	0	0	87.5	23.14	36.6	82.98	19.9	32.1	100	18.85	31.72
Geodo	576	0	0	0	57.23	78.51	66.2	57.75	97.94	72.66	47.45	98.94	64.14
Padcrypt	576	95.54	93.04	94.27	94.29	90.83	92.52	98.1	98.1	98.1	92.54	98.41	95.38
Ramdo	2000	96.93	99.76	98.33	99.18	98.1	98.63	99.83	99.5	99.66	96.95	100	98.45

Table 6 (continued)

Family	#Num	Invincea			MIT			LSTM-MI			TS-ASRCaps		
		P	R	F	P	R	F	P	R	F	P	R	F
Corebot	240	98	100	98.99	100	100	100	100	97.53	98.75	100	97.83	98.9
Volatile	996	98.48	100	99.23	97.96	100	98.97	98.99	100	99.49	95.54	99.48	97.47
Bamital	240	100	81.25	89.66	100	100	100	94.81	100	97.33	100	100	100
Beebone	210	100	100	100	100	100	100	100	98.33	99.16	97.44	100	98.7
Dircrypt	720	0	0	0	0	0	0	0	0	0	0	0	0
Tempedreve	249	0	0	0	66.67	18.18	28.57	58.33	17.28	26.67	0	0	0
Fobber	600	0	0	0	37.18	20.86	26.73	30.88	25.93	28.19	44.44	3.15	5.88
Vidro	200	0	0	0	0	0	0	0	0	0	0	0	0
Micro	–	94.56	94.56	94.56	96.76	96.76	96.76	96.78	96.78	96.78	96.82	96.82	96.82
Macro	–	63.81	55.1	57.37	72.93	66.63	68.12	71.75	68.12	68.46	70.23	66.16	66.11

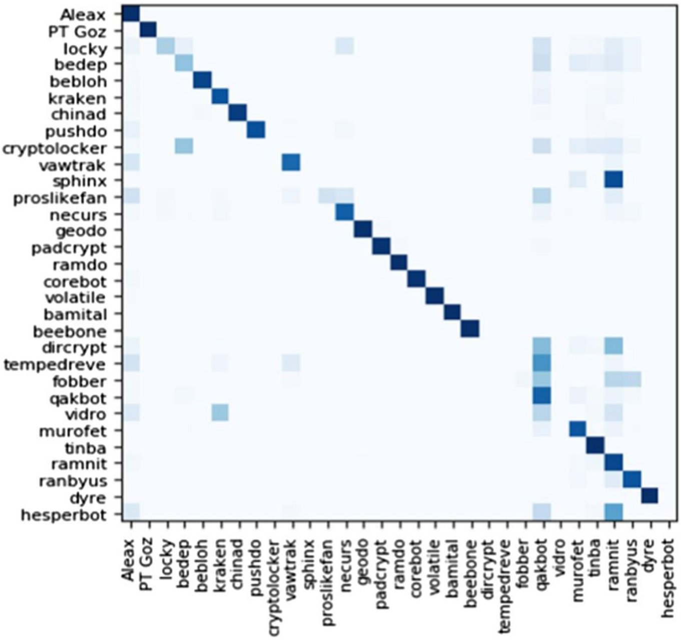


Fig. 6 Normalized confusion matrix of the TS-ASRCaps classifier

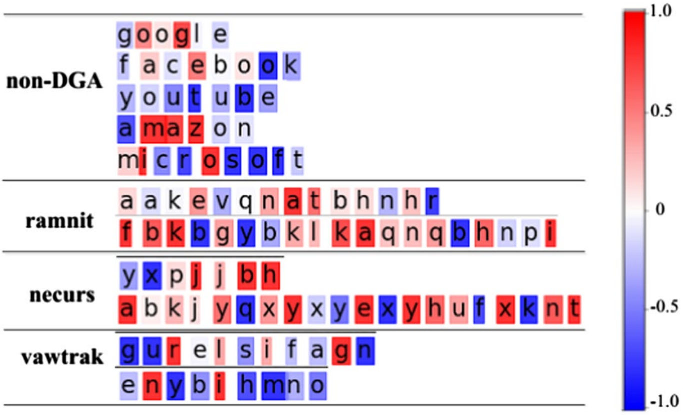


Fig. 7 Examples of the visualization results of the attention matrix

which include the feature extraction and classification time. Figure 8 displays the distribution of overall runtimes for all models. The five-pointed star denotes the average runtime and the segment inside the box shows the median.

For our method, the average lies at just 0.015 s for processing one domain name on a single core (GeForce GTX 1060 with 8 GB RAM), of which 0.0005 s spent in the feature extraction and 0.015 s in the classification. Although our method is

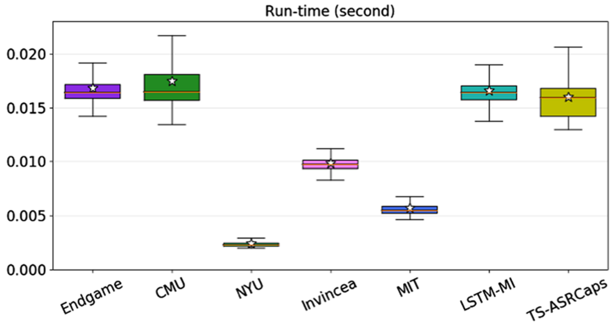


Fig. 8 Frequency distribution of runtimes

slower than that of some existing methods, it has almost no computation cost. Additionally, as shown in Table 3, the baseline methods achieve competitive results on some datasets but fail to adapt to the others. For example, Endgame and LSTM-MI perform quite well on OSINT, Lab360 and AR but poorly on XJU, or Invincea has a favorable performance on OSINT but lower values on other datasets. Our proposed method performs consistently well on all datasets that demonstrates the good generalization ability. Apparently, given the substantially superior performance of the TS-ASRCaps over other state-of-the-art techniques, the additional cost incurred by the TS-ASRCaps can be seen to be justified. Overall, our method allows for computationally inexpensive feature extraction and classification. Based on the above facts, we can claim that the proposed method can facilitate real-time detection of DGAs.

5 Discussions

In this paper, we explored a novel Two-Stream Network to simultaneously capture the semantic distribution and spatial context information contained in DGA domain names, without relying on any other complex or expert features. We evaluated our framework from two aspects: detecting DGA and non-DGA, and distinguishing one DGA algorithm from another. To the best of our knowledge, this is the first application of the multimodal deep learning to the DGA botnet detection.

Though the TS-ASRCaps performed extremely well in our experiments, there might be room for improvement. We will work on extending our system to incorporate new multimodal features into the system, which may promote the representation of higher-level concepts. Another important consideration is the modularity of the system. Future work is needed to replace some of the components contained in this architecture with newer and better-performing versions.

Acknowledgments The authors would like to thank the Editor-in-Chief, the Associate Editor, and the reviewers for their insightful comments and suggestions. This work was supported by the Research Innovation Project of Graduate Student in Xinjiang Uygur Autonomous Region (XJ2019G065), the CERNET Innovation Project (NGII20170420, NGII20190412) and the Xinjiang Uygur Autonomous Region Cyber Security and Informatization Project (XJWX-1-Z-2019-1021).

References

1. Yang L, Liu G, Zhai J, Dai Y, Yan Z, Zhou Y, Huang W. A novel detection method for word-based DGA. International Conference on Cloud Computing and Security, 472–483 (2018)
2. Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Dagon D. From throw-away traffic to bots: detecting the rise of DGA-based malware. Usenix Security Symposium, 491–506 (2012)
3. Krishnan S, Taylor T, Monrose F, Mchugh J. Crossing the threshold: detecting network malfeasance via sequential hypothesis testing. IEEE/IFIP International Conference on Dependable Systems & Networks, 1–12 (2013)
4. Yu B, Gray D L, Pan J, Cock M D, Nascimento A C. Inline DGA detection with deep networks. IEEE International Conference on Data Mining Workshops, 683–692 (2017)
5. Catania C, García S, Torres P. Deep convolutional neural networks for DGA detection. Argentine Congress of Computer Science ,327–340 (2018)
6. Tran, D., Mac, H., Tong, V., Tran, H.A., Nguyen, L.G.: A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing **275**, 2401–2413 (2018)
7. Schiavoni S, Maggi F, Cavallaro L, Zanero S. Phoenix: DGA-based botnet tracking and intelligence. International Conference on detection of intrusions and malware, and vulnerability assessment, 192–211 (2014)
8. Mowbray M, Hagen J. Finding domain-generation algorithms by looking at length distribution. IEEE international symposium on software reliability engineering workshops (2014) 395–400
9. Sivaguru R, Choudhary C, Yu B, Tymchenko V, Nascimento A, Cock M D. An evaluation of DGA classifiers. IEEE International Conference on Big Data, (2018) 5058–5067
10. Li, Y., Xiong, K., Chin, T., Hu, C.: A machine learning framework for domain generation algorithm-based malware detection. IEEE Access **7**, 32765–32782 (2019)
11. Wang Z, Jia Z, Zhang B. A detection scheme for DGA domain names based on SVM. International Conference on mathematics, modelling, simulation and algorithms, (2018)
12. Tong V, Nguyen G. A method for detecting DGA botnet based on semantic and cluster analysis. Seventh Symposium on information and communication technology, 272–277 (2016)
13. Dahal B, Kim Y. AutoEncoded domains with mean activation for DGA botnet detection. IEEE International Conference on global security, safety and sustainability, 208–212 (2019)
14. Luo X, Wang L, Xu Z, Yang J, Sun M, Wang J. Dgasensor: Fast detection for dga-based malwares. International Conference on communications and broadband networking, 47–53 (2017)
15. Koh JJ, Rhodes B. Inline detection of domain generation algorithms with context-sensitive word embeddings. IEEE International Conference on Big Data, 2966–2971 (2018)
16. Yang M, Wen Q. Detecting android malware by applying classification techniques on images patterns. IEEE International Conference on cloud computing and big data analysis, 344–347 (2017)
17. Su J, Vasconcellos V D, Prasad S, Daniele S, Feng Y, Sakurai K. Lightweight classification of IoT malware based on image recognition. IEEE Annual computer software and applications conference, 664–669 (2018)
18. Dey A, Bhattacharya S, Chaki N. Byte label malware classification using image entropy. Advanced computing and systems for security, 17–29 (2019)
19. Yen, Y.S., Sun, H.M.: An android mutation malware detection based on deep learning using visualization of importance from codes. Microelectron Reliab. **93**, 109–114 (2019)
20. Li S, Li W, Cook C, Zhu C, Gao, Y. Independently recurrent neural network (indrnn): building a longer and deeper rnn. IEEE Conference on computer vision and pattern recognition, 5457–5466 (2018)
21. Li B, Cheng Z, Xu Z, Ye W. Long text analysis using sliced recurrent neural networks with breaking point information enrichment. IEEE International Conference on acoustics, speech and signal processing, 7550–7554 (2019)
22. Yu Z, Liu G. Sliced recurrent neural networks. International Conference on computational linguistics, 2953–2964 (2018)
23. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. International Conference on learning representations (2014)
24. Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules. Advances in neural information processing systems, 3856–3866 (2017)
25. Wang S, Zhou G, Lu J, Zhang F. A Novel Malware Detection and Classification Method Based on Capsule Network. International Conference on artificial intelligence and security, 573–584 (2019)

26. Kim, J., Jang, S., Park, E., Choi, S.: Text classification using capsules. Neurocomputing **376**, 214–221 (2020)
27. "Does Alexa have a list of its top-ranked websites?" Amazon. https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites-. Accessed 20 July 2019
28. "OSINT feeds from Bambenek consulting," Bambenek Consulting. http://osint.bambenekconsulting.com/feeds/. Accessed 20 July 2019.
29. Lab, accessed: 2019-07–20. . https://data.netlab.360.com/dga/
30. Abakumov A. https://github.com/andrewaeva/DGA. Accessed 20 July 2019
31. Yu B, Pan J, Hu J, Nascimento A, Cock M D. Character level based detection of DGA domain names. International Joint Conference on neural networks, 1–8 (2018)
32. Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. Advances in neural information processing systems, 649–657 (2015)
33. Vosoughi S, Vijayaraghavan P, Roy D. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. International ACM SIGIR Conference on research and development in information retrieva, 1041–1044 (2016)

Xinjun Pei Was born in Tacheng, Xinjiang, China, in 1995. He is currently pursuing the M.S. degree with the School of Information Science and Engineering, Xinjiang University, Urumqi, China. Since 2016, he has been engaged in the direction of information security. His research interests include deep learning and mobile security.

Shengwei Tian Was born in Urumqi, Xinjiang, China, in 1973. He received the B.S., M.S., and Ph.D. degrees from the School of Information Science and Engineering, Xinjiang University, Urumqi, China, in 1997, 2004 and 2010, respectively. Since 2002, he has been a Teacher with the School of Software, Xinjiang University, where he is currently a Professor. His research interests include artificial intelligence, natural language processing, and cyberspace security.

Long Yu Was born in Urumqi, Xinjiang, China, in 1973. She received the B.S. and M.S. degrees from the School of Information Science and Engineering, Xinjiang University, Urumqi, China, in 1997 and 2008, respectively. Since 2002, she has been a Teacher with the School of Information Science and Engineering, Xinjiang University, where she is currently a Professor. Her research interests include data mining, mobile security, and cyberspace security.

Huanhuan Wang Was born in Shangqiu, Henan, China, in 1995. She is currently pursuing the M.S. degree with the School of Software, Xinjiang University, Urumqi, China. Her research interests include deep learning and mobile security.

Yongfang Peng Was born in Zaozhuang, Shangdong, China, in 1994. She is currently pursuing the M.S. degree with the School of Software, Xinjiang University, Urumqi, China. Her research interests include deep learning and mobile security.

Affiliations

Xinjun Pei[1] · Shengwei Tian[2] · Long Yu[3] · Huanhuan Wang[2] · Yongfang Peng[2]

Xinjun Pei
pei_xinjun@163.com

Long Yu
yul@xju.edu.cn

Huanhuan Wang
1597628677@qq.com

Yongfang Peng
m13999412597@163.com

[1] School of Information Science and Engineering, Xinjiang University, Urumqi 830001, Xinjiang, China

[2] School of Software, Xinjiang University, Urumqi 830001, Xinjiang, China

[3] Network Center, Xinjiang University, Urumqi 830001, Xinjiang, China