A Multi-Dimensional Deep Learning Framework for IoT Malware Classification and Family Attribution

Mirabelle Dib, *Member, IEEE*, Sadegh Torabi, *Member, IEEE*, Elias Bou-Harb, *Senior Member, IEEE*, and Chadi Assi, *Fellow, IEEE*

***Abstract*—The emergence of Internet of Things malware, which leverages exploited IoT devices to perform large-scale cyber attacks (e.g., Mirai botnet), is considered as a major threat to the Internet ecosystem. To mitigate such threat, there is an utmost need for effective IoT malware classification and family attribution, which provide essential steps towards initiating attack mitigation/prevention countermeasures. In this paper, motivated by the lack of sophisticated malware obfuscation in the implementation of IoT malware, we utilize features extracted from strings- and image-based representations of the executable binaries to propose a novel multi-dimensional classification approach using Deep Learning (DL) architectures. To this end, we analyze more than 70,000 recently detected IoT malware samples. Our in-depth experiments with four prominent IoT malware families highlight the significant accuracy of the approach (99.78%), which outperforms conventional single-level classifiers. Additionally, we utilize our IoT-tailored approach for labeling newly detected "unknown" malware samples, which were mainly attributed to a few predominant families. Finally, this work contributes to the security of future networks (e.g., 5G) through the implementation of effective tools/techniques for timely IoT malware classification, and attack mitigation.**

***Index Terms*—IoT malware classification, deep learning, multimodal learning, feature fusion, static malware analysis.**

I. Introduction

INTERNET of Things (IoT) devices have been integrated in different aspects of our everyday activities. Despite their benefits, the rising number of IoT-tailored malware, which aim at utilizing compromised IoT devices (e.g., weak authentication) towards coordinating large-scale cyber attacks, has posed a major threat to the overall Internet ecosystem [1], [2]. For instance, the Mirai botnet was leveraged in the famous cyber attack on Dyn (major U.S. DNS service provider) in October 2016 [1], resulting in one of the largest recorded DDoS attacks on the Internet. More importantly, the release of the Mirai source code fueled the rapid evolution of more advanced and sophisticated Mirai-like malware such as Hajime [3], Satori [4], and BrickerBot [5], to name a few.

It is imperative to evaluate the security of the IoT paradigm as well as develop rigorous mitigation approaches against the spread of IoT malware. These tasks are challenging in the context of IoT due to the lack of empirical data about existing IoT malware and the lack of knowledge about the behavioral characteristics of malware-infected IoT devices. To overcome these challenges, a number of IoT specialized honeypots have been deployed to obtain detailed information about existing IoT malware, including the malware executable/binary [6], [7]. Additionally, static malware analysis techniques can be used to build a better understanding about IoT malware, while extracting features that can improve attack mitigation by developing efficient malware classification techniques using machine/deep learning algorithms. For instance, a number of strings-based features have been utilized to devise ML/DL methods for malware classification/clustering [8]–[10]. Moreover, image-based techniques, which extract features from the image representation of malware binaries, have been effectively used in different contexts [11]–[15]. Finally, models that leverage a combination of features (e.g., CFGs, statistical features, etc.) from different malware characteristics have been proposed for malware classification [9], [12], [16]–[19].

To this end, we propose a multi-level approach that leverages a combination of static features along with DL techniques for effective IoT malware classification and family attribution. Our objective is threefold: (i) to leverage IoT-specific properties such as the lack of widely deployed sophisticated malware obfuscation [20], [21] for extracting static features from different representation of the IoT malware binaries that are not empirically feasible in other contexts (e.g., widely obfuscated Windows PE or Android malware) [22], (ii) to leverage deep learning methods capabilities to automatically extract static features without relying on expensive feature-engineering, and (iii) improving the overall classification accuracy by building a multi-dimensional DL architecture that utilizes different representations of the target IoT malware binaries.

To achieve our objectives, we leverage about 70,000 real instances of IoT malware binaries/executables obtained from VirusTotal, VirusShare, and a specialized IoT honeypot (IoTPOT [6]) over a period of 20 months (September

Manuscript received October 9, 2020; revised March 1, 2021; accepted April 19, 2021. Date of publication April 23, 2021; date of current version June 10, 2021. This work has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Concordia University. This work was also partially supported by a grant from the U.S. National Science Foundation (NSF), Office of Advanced Cyberinfrastructure (OAC), #1907821. The associate editor coordinating the review of this article and approving it for publication was C. Fung. *(Corresponding author: Chadi Assi.)*

Mirabelle Dib, Sadegh Torabi, and Chadi Assi are with the Cyber Security Research Centre, Concordia Institute for Information Systems Engineering, Montreal, QC H3G 1M8, Canada (e-mail: mi_dib@encs.concordia.ca; sa_tora@encs.concordia.ca; assi@encs.concordia.ca).

Elias Bou-Harb is with the Cyber Center for Security and Analytics, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: elias.bouharb@utsa.edu).

Digital Object Identifier 10.1109/TNSM.2021.3075315

2018–May 2020). We utilize AVClass [23] to investigate IoT malware family labels as perceived from VirusTotal reports while identifying about 26,000 IoT malware samples that were "unknown" or "unseen" by major antivirus vendors. Additionally, motivated by the lack of malware obfuscation in the IoT context, we devise string- and image-based analysis techniques using convolutional neural network (CNN) and long short-term memory recurrent neural network (LSTM), respectively. Consequently, we implement our multi-level DL architecture by fusing the learned features from each subcomponent through a neural network classifier. Finally, we perform a series of experiments using about 10,000 IoT malware samples from four different predominant families and evaluate the effectiveness of our approach in comparison to state-of-the-art approaches that implement single-level strings-based and/or image-based classifiers.

In general, our experimental analysis demonstrates the effectiveness of our purposed multi-level approach towards classifying IoT malware families. Indeed, the implemented classifier produces a significantly improved accuracy (99.78%), which outperforms state-of-the-are single level classifier implementations (accuracy$\leq$95%). In addition to classifying known IoT malware, we leverage the proposed approach for classifying unknown/unseen IoT malware samples, which are attributed to a few prominent IoT malware families (e.g., Mirai). Finally, while we empirically evaluate the effectiveness of the proposed approach for IoT malware classification, we discuss future work towards IoT malware threat mitigation using multi-level classification techniques.

To this end, this work makes the following main contributions:

- Given the challenges associated with IoT malware classification and family attribution, in this paper, we are among the first to introduce a holistic, multi-level approach for analyzing IoT malware by combining the benefits of static malware analysis with deep learning classification techniques. More importantly, despite the fact that IoT malware families tend to be similar in terms of implementation and overall behaviors [20], our results show that the proposed multi-level approach can be used to perform effective and efficient classification by considering granular characteristics from the analyzed malware binaries.
- We implement the proposed approach by utilizing DL methods to automatically extract static-based features to overcome the challenges associated with feature-engineering methods. Moreover, we evaluate the multi-level deep learning model with 10,234 recently collected IoT malware executable binaries. The results indicate a significantly improved classification accuracy (accuracy = 99.78% and F1-score = 99.57%), as compared to classifiers that rely on a single modality of data.
- To the best of our knowledge, we are among the first to obtain and analyze a large and representative sample of real IoT malware executables, which contains a variety of IoT malware variants detected in recent years. While our analysis results indicate the effectiveness of the proposed classification approach for attributing malware samples to known families, we also leverage the multi-level classifier to predict the labels of 24,271 unknown malware samples that have not been detected/labeled by major AV vendors. Moreover, while our extended strings-based similarity analysis corroborates the labeling outcomes, we uncover indications of new Mirai variants related to the Covid-19 pandemic, which highlights the rapid evolution of IoT malware found in the wild.

The rest of the paper is organized as follows. Section II details the methodology of our proposed multi-level framework. The experimental classification and prediction results are presented in Section III. We discuss the main outcomes of this work, as well as limitations and future research directions in Section V. We present a detailed literature review in Section IV, followed by concluding remarks in Section VI.

II. Methodology

In this paper, we attempt to address the IoT malware classification and family attribution problem through analyzing IoT malware binaries collected in the wild (see Section II-A). While using AI-based techniques for malware classification is not a new problem, previous works mainly rely on the use of static/dynamic analysis techniques along with ML-based classifiers, which often require domain knowledge and costly feature extraction/evaluation operations. More importantly, with the continuous evolution of IoT malware, such handcrafted features might not be useful for detecting/classifying emerging malware families. Hence, recent work using DL-based malware classification approaches have been proposed to overcome such challenges by reducing the cost of artificial feature engineering while learning features directly from raw data without the need for additional feature engineering. Nevertheless, most of the state-of-the-art approaches still rely on a single representation of the malware data to learn static or dynamic features, which limits the learning process while ignoring the benefits of using other representations of the target data.

To address the above mentioned limitations, in this paper, we propose a multi-dimensional DL malware classification approach that can detect and attribute malware executable binaries to known IoT malware families. Our aim is to develop and evaluate a classification approach that automatically learns static features from multiple representation of the malware binary, while improving the overall classification accuracy through combining multiple modalities. In particular, we attempt to answer the following research questions (RQs):

1) How can we utilize static malware analysis techniques to develop an effective multi-level classifier for IoT malware family attribution? Does a multi-level deep-learning approach that combines learned characteristics of malware from various data representation yield a higher classification performance compared to single modality DL algorithms?
2) How can we benefit from next-generation malware analysis techniques to overcome the challenges associated with feature-engineering? How effective is the proposed multi-level deep learning approach as compared to state-of-the-art ML approaches that utilize

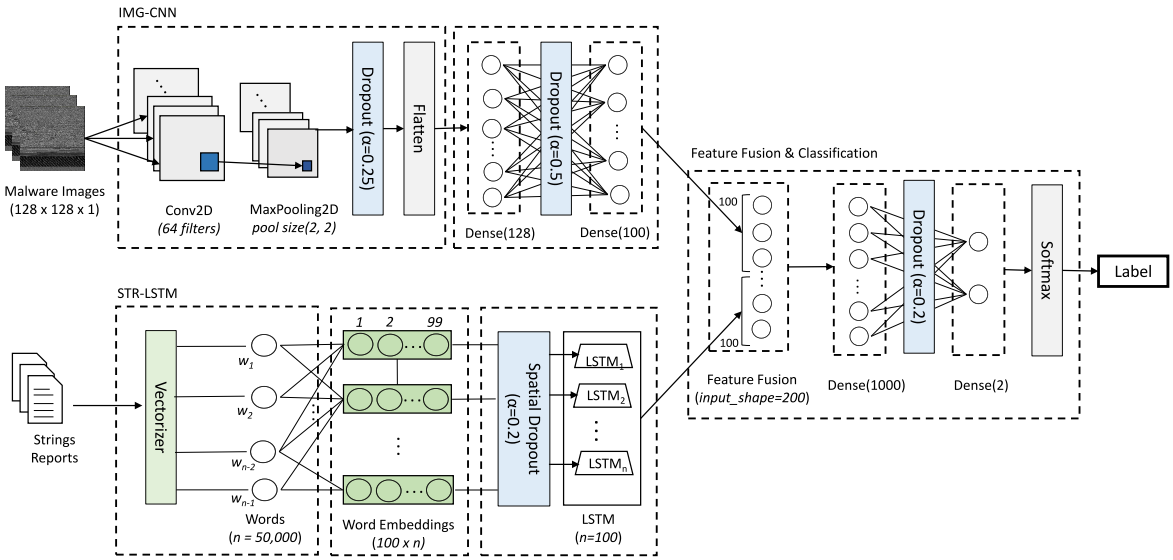


Fig. 1. Overview of the multi-level deep learning malware classification system.

various combinations of features and feature engineering techniques?

3) How to leverage the developed multi-level classifier to detect new or unknown malware samples given the information about existing malware families?

The architecture of our multi-level deep learning framework for IoT malware classification is shown in Figure 1. In the proposed framework, the input is an ELF executable binary for Linux-based systems, while the classification outcome represent the IoT malware family label. The classification module consists of 3 main components: (1) image-based component, (2) string-based component, and (3) the feature fusion and classification component. The strings- and image-based components extract/learn corresponding features from different representation of the malware. Consequently, the final component is responsible for fusing the learned features into a shared representation, which is used to produce the final classification outcome. A complete description of the dataset used for evaluation, as well as an in-depth analysis of the different sub-components and features types chosen, are provided in the next sections.

A. IoT Malware Collection and Family Labeling

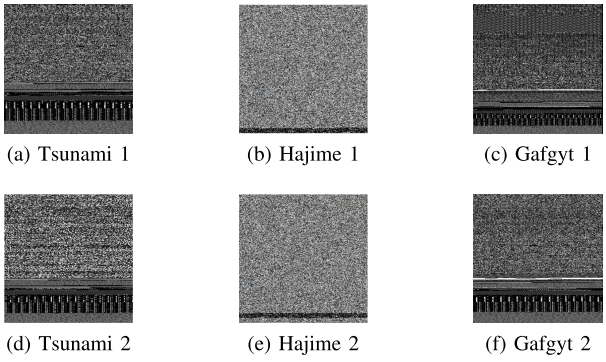
In this paper, we leverage well-recognized online malware repositories such as VirusTotal and VirusShare along with a specialized IoT honeypot (IoTPOT [6]) to obtain a total of 91,776 samples collected between *2018-09-14* and *2020-05-25*. To ensure consistency, we performed pre-processing steps to filter out corrupted or non-executable files (e.g., HTML/ASCII files) ending up with 74,429 IoT malware binaries. The dataset represents 18 different malware families, which were labelled by using AVClass [23] and malware analysis reports from VirusTotal (VT) [24]. AVClass performs malware labeling by applying a majority rule on reported malware labels from multiple anti-virus vendors, as perceived on VirusTotal reports.

As summarized in Table I, the majority of the detected IoT malware are classified as Mirai (about 55%), followed by a significantly fewer number of samples labeled as Gafgyt (about 5%), and Tsunami (about 1.3%), respectively. Furthermore, some samples were not assigned with any family name/label, or were associated with a generic malware label (e.g., Linux). We label such samples as "Unknown" throughout the work. In addition, we identify 24,271 malware sample that have not been detected by antivirus vendors (labeled as "Unseen"), as perceived from VirusTotal reports. Finally, it is important to realize that all the IoT malware data used in this work is available for research purposes and can be directly requested from the aforementioned sources (e.g., IoTPOT). Nevertheless, due to restricted sharing policies specified by the data providers, we are unable to share the analyzed data directly with the research community.

TABLE I
DISTRIBUTION OF MALWARE BY FAMILY

Malware Family	Count	(%)
Mirai	40,974	(55.05)
Gafgyt	3,976	(5.34)
Tsunami	956	(1.28)
Dofloo	464	(0.62)
Others	122	(0.16)
Unknown	2,664	(2.23)
Unseen	24,271	(32.60)
Total	74,429	(100)

Malware Detection Timeline: To verify that our malware labels are accurate and reliable, we considered a 5 months time period between the end of our malware data collection (May 2020) and our performed malware family labelling in October 2020 to avoid collecting incomplete or inaccurate malware family labels. Specifically, at the time when this research was conducted (October 2020), the 24,271 identified unknown malware samples were never seen on VirusTotal reports even after 5 months from being detected by IoTPOT. This affirms



(a) Tsunami 1 (b) Hajime 1 (c) Gafgyt 1

(d) Tsunami 2 (e) Hajime 2 (f) Gafgyt 2

Fig. 2. Grayscale images of malware samples belonging to the Tsunami, Hajime and Gafgyt Families.

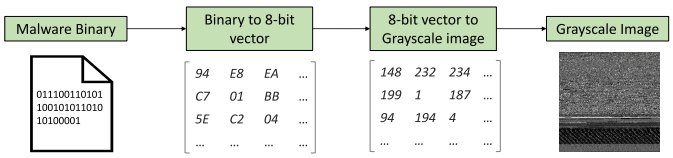


Fig. 3. Process of visualizing a malware as a grayscale image.

the effectiveness and ability of specialized IoT honeypots to promptly detect IoT-tailored malware.

B. Image-Based Component

In this paper, we use the approach proposed by Nataraj *et al.* [11] to visualize malware binary files as grayscale images. In particular, a malware binary can be read byte-by-byte as a vector of 8 bit unsigned integers and then organized into a 2D array (Figure 3). This can be visualized as a gray scale image whose pixel values range from 0 to 255 (0: black, 255: white). As illustrated in Figures 2(a–f), we can clearly observe the high resemblance between the image representation of two different malware samples that belong to the same malware family, respectively.

The literature has demonstrated the effectiveness of malware classification techniques using malware image representation. Therefore, as depicted in Figure 1, the image-based component takes the bytes representation of a malware grayscale image as input. In our specific implementation, we adopt CNN and LSTM neural networks for our image-based classification algorithms. Additionally, we perform hyper-parameter tuning to select the best combinations of hyper-parameters. We evaluate the performance of both algorithms and present the optimal architecture, which was selected for our final multi-level model in Section III-A, respectively.

C. String-Based Component

In addition to the image representation, we utilize reverse-engineering techniques to extract meaningful strings from the binary code. Specifically, we utilize the *strings* utility in Linux to extract printable strings with three or more bytes. Additionally, we are interested in identifying strings that exhibit similar contextual information, which can help towards grouping IoT malware samples according to the occurrences of the same strings. Such embedded strings can provide

```
rm -rf %s;
pkill -9 %s; killall -9 %s;
cd /tmp || cd /var/run || cd /dev/shm || cd /mnt || cd /var;
rm -f *; /bin/busybox wget http://185.*.*.*/love; sh love; wget
http://185.*.*.*/love; sh love; /bin/busybox tftp -r tftp.sh -g
185.*.*.*; sh tftp.sh; /bin/busybox tftp -c get tftp2.sh -g
185.*.*.*; sh tftp2.sh
```

Fig. 4. Example of readable strings extracted from a malware sample.

clues about the suspect malware and its functionalities (e.g., attack commands, IP addresses, filenames, unique strings, etc.). For instance, the example of extracted strings illustrated in Figure 4 shows that the analyzed malware is trying to download and execute a malicious file from a possible adversarial C&C server.

Data Representation: Given the extracted strings files, we utilize natural language processing (NLP) techniques to tokenize the top 50,000 most common words in each file. A naive approach to convert words to vectors is to assign each word with a "one-hot vector", which means that the vector would be all zeros except one unique index for each word. However, this type of word representation can introduce substantial data sparsity. Instead, we adopt a continuous vector space representation of the extracted words (embeddings) that allows semantically similar words to be mapped to nearby points, thus encoding useful information about the words' actual meaning/use in the text. Word embeddings are usually joint with neural network models for document classification. Accordingly, we implement CNN and LSTM models for our text/strings classification. Consequently, we perform 10-fold cross validation to evaluate/compare the effectiveness of the implemented models. The optimal architecture, which was configured following a grid search over the hyper-parameters of the networks, is presented in Section III-B.

Malware Obfuscation: It is a common technique deployed by malware writers to hide all or parts of their implementation while avoiding rapid analysis and detection by conventional static and signature-based techniques [25]. We leverage FLOSS [26] tool to investigate the presence of obfuscated malware strings in the analyzed samples. Interestingly, we found that the obfuscated IoT samples were mainly packed by off-the-shelf tools such as UPX [27], which can be easily de-obfuscated. Indeed, our analysis confirms the lack of sophisticated malware obfuscation in IoT by extracting and de-obfuscating strings from the majority (about 76%) of the IoT binaries. To maintain consistency, we decide to use malware samples that were successfully de-obfuscated throughout our analysis, while discarding the remaining samples from further analysis. Note that although our proposed approach does not consider obfuscated malware binaries, it is still effective in the context of IoT, where it can be leveraged to analyze a significant portion of the detected IoT malware samples in the wild.

D. Fusion Component and Classification

As illustrated in Figure 1, each sub-component in our framework extracts features from a different representation of malware, i.e., a different data modality. The fusion component is responsible for combining the learned features from

multiple sub-component into a shared representation, which is used to enable final classification outcomes. It this work, we aim at demonstrating the effectiveness of applying intelligent fusion of different modalities of features to achieve better classification outcomes, as compared to using single-level classifiers.

To achieve this, we perform per-component pre-training before the final feature fusion and classification. This step is done to avoid overfitting a subset of features that belong to one data modality over the others [17]. Additionally, we pre-trained each component separately while optimizing their hyper-parameters to initialize each component in the multimodal neural network with the optimal learned pre-trained weights, respectively. This is an idea borrowed from transfer learning and feature fusion, where the knowledge and the features learned by each model are transferred into the multimodal neural network to save training time, while converging faster towards better classification results [17], [28]. During the process of transferring knowledge, the following important questions must be answered.

What to Transfer & When to Transfer: We must comprehend which part of the knowledge learned can be transferred from the source to the target in order to improve the performance of the target task. We should aim at utilizing feature fusion to improve the target classification results and not degrade them. For that reason, we must first pre-train our sub-component deep learning models and record their performance, and then proceed towards fusing the most relevant learned features by each sub-component to compare the classification results with the multi-level model.

How to Transfer: Once the first two questions above have been answered, we can proceed towards identifying ways of transferring the knowledge across different data modalities. Deep learning models are layered architectures that learn various features at different layers. All the layers are finally connected to a *fully connected* layer that is responsible for generating the final output. The key idea of transferring knowledge is to leverage the pre-trained models' weighted layers to extract features, and abandon the models' final classification layer. Hence, each sub-component will act as a feature extractor for the final multi-level deep learning model, which will fuse the different features learned into a joint multi-modal representation

Feature Fusion and Classification: As shown in Figure 1, the learned representations *I* of the image-based component and *S* of the string-based component are continually generated across multiple fully connected layers during the training phase. Consequently, the features learned by the last fully connected layer of each sub-component are fused into a shared multi-modal representation at the final phase. Note that vector *I* of size *i* and vector *S* of size *s* are fused into a vector *M* of size *m*, where $m = i + s$. This joint multi-modal representation *M* is then fed into a neural network with two fully connected layers and a dropout layer. The last fully connected layer is responsible for classifying a malicious binary as follows: $prediction = softmax(b_c + W_c P)$, where prediction is a vector of size *C* (number of classes), and W_c and b_c are the weights and biases of the layer. The *softmax* function outputs the probability of a malware to belong to any of the malware families in the training set. The sizes of vectors *I* and *S* are determined during the configuration of the network. Hyperparameter optimization is performed accordingly to set the numbers of hidden units for yielding the best results (see Section III). It is important to realise that combining a larger number of features in the final layer will always result in a higher dimensional feature set, which will negatively affect the overall classification outcomes.

Model Evaluation: To compare the effectiveness of the deployed models, we rely on standard machine learning measures such as accuracy, precision, recall, and F1-score. Precision is the ratio of correctly classified IoT malware samples over all the IoT malware samples designated as such ($precision = [t_p/(t_p + f_p)]$). The recall is the ratio of correctly classified IoT malware samples over the total number actually existing in the test data ($recall = [t_p/(t_p + f_n)]$). While the precision allows the model to designate only the actual relevant samples as relevant, the recall validates the model's ability to find all relevant samples within a given dataset. The F1-score combines these two metrics together by taking the weighted average (i.e., the harmonic mean) of precision and recall ($F1 = 2.(precision.recall/precision + recall)$). Finally, we consider the best model according to the macro F1-score, because the accuracy measure by itself might be misleading when used with imbalanced data, whereas the macro F1-score metric gives more importance to False Negatives and False Positives.

III. Experimental Results

In this section, we use empirical data to evaluate the effectiveness of the implemented classification approach, while comparing its outcomes to the single-level modality approaches and the state-of-the-art ML techniques with feature engineering. To develop our proposed multi-level malware classification model, we use Keras API to implement the corresponding image- and strings-based components using both CNN and LSTM models. The objective is to evaluate the effectiveness of two different implementation of each components using CNN and LSTM, while choosing the final model that produces the best accuracy and F1-score outcomes.

Additionally, to address the problem of class imbalance within the training dataset, we apply data resampling to obtain 10,234 malware samples representing four prominent IoT malware families: Mirai (5,927), Gafgyt (3,227), Tsunami (776), and Dofloo (304). Moreover, we scale the calculated loss for each observation in the models by the appropriate class weight to assign more significance to the losses associated with the minority classes [29], [30]. To validate the stability and generalizability of the deployed models to an independent (unknown) dataset, we leverage a stratified version of k-fold cross validation ($k = 10$). We calculate the average model score across multiple validation iterations while preserving the class distribution in the train and test sets for each evaluation of a given model. Accordingly, the dataset, which consists of 10,234 malware samples, is divided into *k* subsets, where the models are trained with *k*–1 subsets and tested with the last subset over *k* iterations. Further, to select the

TABLE II
HYPER-PARAMETER TUNING/SELECTION AND 10-FOLD CROSS VALIDATION PERFORMANCE EVALUATION RESULTS FOR THE SELECTED DL MODELS

Parameters	Space	IMG-CNN	STR-CNN	IMG-LSTM	STR-LSTM	Multi-Level Model
Num. of filters (f)	32,64	64	32	-	-	-
Num. of units (u)	2,50,100,128,1000	128, 100	-	128	100	1000, 4
Kernel size (w x w)	(2,2),(3,3),(4,4)	(3,3)	(4,4)	-	-	-
Pool size (p)	2,3	2	2	-	-	-
Batch size	32,64,128	64	64	64	64	64
Epochs	10,15,20,30	15	10	10	10	10
Activations	Relu, Elu	Relu	Relu	Relu	-	Relu
Dropout	(0, 0.2, 0.25, 0.3)	0.25	0.5	0.2	0.2	0.3
Spatial droput	(0, 0.1, 0.2)	-	-	-	0.2	-
Validation split	(0.1, 0.2, 0.25, 0.3)	0.2	0.2	0.25	0.2	0.2
Accuracy	–	0.9722	0.9886	0.9711	0.9840	**0.9978**
Macro F1 score	–	0.9721	0.9851	0.9702	0.9820	**0.9957**

best model implementation, we perform grid search using Talos [31], which automates the hyper-parameters tuning and model evaluation processes. The optimized hyper-parameters are presented in Table II.

A. Evaluating the Image-Based Component

We present the analysis of the performance of two deep learning algorithms, CNN and LSTM, that we designed to classify IoT malware based on their bytes-based image representation. As shown in Table II, the results of the 10-fold cross validation on the implemented models illustrate that both models perform significantly well with high accuracy and F1-score outcomes. However, the CNN implementation of the image-based components yields slightly better outcomes, with about 97.2% for both the classification accuracy and macro F1-score. Note that the optimal architecture of the CNN was configured after a applying grid search over the hyper-parameters of the network, as summarized by the results in Table II. The final CNN model, which represents the image-based component in our proposed multi-level IoT malware classification approach (Figure 1), consists of the following layers:

- *Input Layer:* The input of the network is a 128*128*1 image array of pixel values in the range [0 – 255].
- *Convolutional Layer:* The main building block of a CNN is the convolutional layer. It is responsible for applying various convolution filters (64) over the pixels to produce a feature map. Each convolution filter has specific height and width, in our case, 3 x 3, and by design it covers the entire depth of its input. The final output of the convolution layer is a distinct feature map, put together by stacking all feature maps from multiple convolutions on the input. The activation function adopted is the *relu* function [32].
- *Pooling Layer:* After a convolution operation, *pooling* is performed to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and fights overfitting. We apply *max pooling* which slides a window of size 2*2 over its input, and simply takes the max value in the window. After the pooling layer, we perform a *dropout* of 2.5% to prevent overfitting. which makes the network perform better.
- *Fully-Connected Layer:* After the convolution and pooling layers, we add fully connected layers to wrap up the CNN architecture. The output of both convolution and pooling layers are 3D volumes. Since a fully connected layer expects a 1D vector of numbers, we *flatten* the output of the final pooling layer to a vector, which becomes the input to the fully connected layer. Our first fully connected layer consists of 128 units, followed by a dropout of 0.5 and a second fully connected layer with 100 units. Our last fully connected layer combines the features learned by the previous layers and applies the *softmax* function to output the normalized probability distribution over malware families.

B. Evaluating the String-Based Component

We compare the performance of the CNN and LSTM deep learning models to classify IoT malware based on the strings extracted from the malware. As presented in Table II, the 10-fold cross validation and evaluation results of the two algorithms demonstrate significant accuracy and F1-scores for both implementations (about 98%). Nevertheless, it can be observed that the CNN model implementation produced a relatively higher accuracy (98.86%) and macro F1-score (98.51%), thus, chosen as our candidate model for implementing the strings-based component within proposed IoT malware classification model (Figure 1). The optimal architecture of the CNN string-based component, which was configured after a grid search over the hyper-parameters of the network (Table II), consists of the following layers.

- *Embedding Layer:* This layer is defined as the first hidden layer of the network. It requires that the input data be integer encoded, so that each word is represented by a unique integer. It takes as arguments the input dimension, i.e., the size of the vocabulary set to 50,000 words in our case, the output dimension, i.e., the size of the vector space in which words will be embedded (100), and the input length, i.e., input sequences that have 400 words each.
- *Convolutional Layer:* The CNN's convolutional layer "scans" text which is organized into a matrix, with each row representing a word embedding, like it would an image, breaks it down into features, and judges whether each feature matches the relevant label or not. The chosen kernel size is 4, and the number of convolutional filters

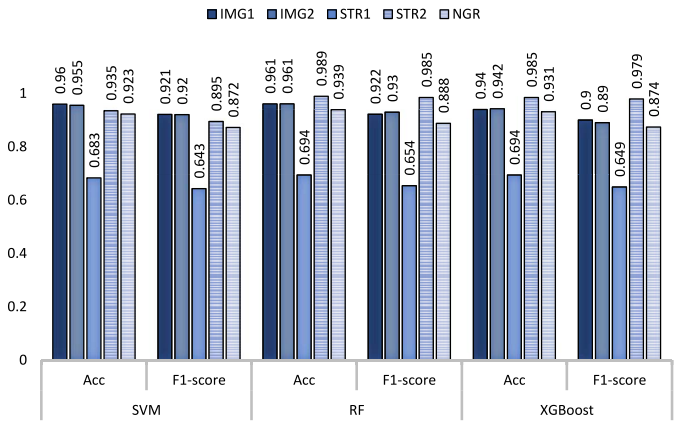


Fig. 5. Evaluation results for the selected feature-engineering approaches.

applied is 32. The activation function adopted is the *relu* function [32].

- *Pooling Layer:* A pooling of size 2 is applied to the input. The pooling stage reduces the dimensionality of the word features and retains only a simple probability score that reflects how likely they are to match a label.
- *Fully-Connected Layer:* At the final stage, these scores, flattened, are the inputs to a fully connected neural layer. The "fully connected" part of the CNN network goes through its own back-propagation process, to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label. The activation function is *softmax* for multi-class classification.

C. Effectiveness of the Proposed Multi-Level DL Model

To answer our RQ1, we evaluate the effectiveness of the proposed multi-level deep learning model against the implemented image- and strings-based components, which are trained on each data modality independently. To do this, we leverage the implemented models for the image- and strings-based components to deploy our multi-level classifier and evaluate its effectiveness. We pre-train each component separately while utilizing their top learned features ($n = 100$ each) as an input to the feature fusion and classification component, as illustrated in Figure 1. The outcomes of the feature fusion and classification steps demonstrate the effectiveness of our multi-level IoT malware classification model with significantly high accuracy and F1-score that exceed 99.5%. Additionally, it is clearly observed that the multi-level model outperforms the DL implementation of the image- and string-based components (Table II). Thus, answering our first research question by demonstrating that the multi-level DL model that learns and combines characteristics of malware from various sources can yield better classification outcomes as compared to DL classifiers that rely on a single modality of data.

D. Comparison With Feature Engineering Approaches

To answer our second research question (RQ2), we perform image- and string-based classification using a set of diverse features that were extracted from the malware grayscale images and strings. While there are many possible combination of features and classification models, for the sake of comparison to our deep learning approaches, we reviewed the state-of-the-art malware classification approaches from the literature [8], [12], [13], [33] and identified combinations of features/models that were more relevant to our experiments.

Image-Based Features: A malware binary is first converted to an image representation, as explained in Section II-B, on which texture based features can be used as visual signatures for each malware family. We extract two sets of features from the grayscale image representation of malware.

- *Haralick* features (**IMG1**), which compute a global representation of texture based on the Gray Level Co-occurrence Matrix, (GLCM), a matrix that counts the co-occurrence of neighboring gray levels in the image [33].
- *Local Binary Pattern* features (**IMG2**) instead, compute a local representation of texture which is constructed by comparing each pixel with its surrounding neighborhood of pixels.

String-Based Features: The combination of extracted printable strings from malware samples forms a kind of "digital fingerprint' for each malware family. We leverage the following feature sets for our comparison approach:

- Histograms related to the frequency distribution of length of strings (**STR1**) among different malware samples.
- *Bag-of-words* (**STR2**), by generating a global list of all of the strings that occur that are more than three bytes and their frequency.
- *N-grams* (**NGR**), where occurrences of n pairs ($n = 2$) of consecutive strings are counted.

Implemented Classifiers: To perform the classification using the aforementioned features, we use Support Vector Machine (SVM), Random Forest (RF) and XGBoost, which have been proven to be consistently effective for implementing image- and string-based classifiers with feature-engineering [8], [12], [13]. The performance results of our adopted feature-engineering approaches after a 10-fold cross validation are presented in Figure 5. Overall, STR2 features resulted in the highest classification outcomes across the deployed models, with an accuracy of about 98.9% and 98.5% when using the RF and XGBoost classifiers, respectively. Despite such promising results, our implemented multi-level DL architecture outperforms all the tested implementations in terms of the overall classification accuracy and F1-score (Table II). More importantly, using feature-engineering comes with practical limitations, which hamper the overall usability and performance of such approaches as compared to the end-to-end DL methods. For instance, the size of the *Local Binary Pattern* features, which increase exponentially with the number of neighbors, and the high dimensionality of the GLCM matrix used to extract *Haralick* features, lead to an increase of computational complexity in terms of time and space. Additionally, a drawback of *bag-of-words* and *n-grams* feature-engineering approaches is that they lead to a high dimensional feature vector due to the large size of strings vocabulary. Besides being tedious and time-consuming, manual feature engineering is problem-specific, error-prone, and limited by human expertise and capabilities. Thus, we answer our second research question by demonstrating that our multi-level DL approach is in

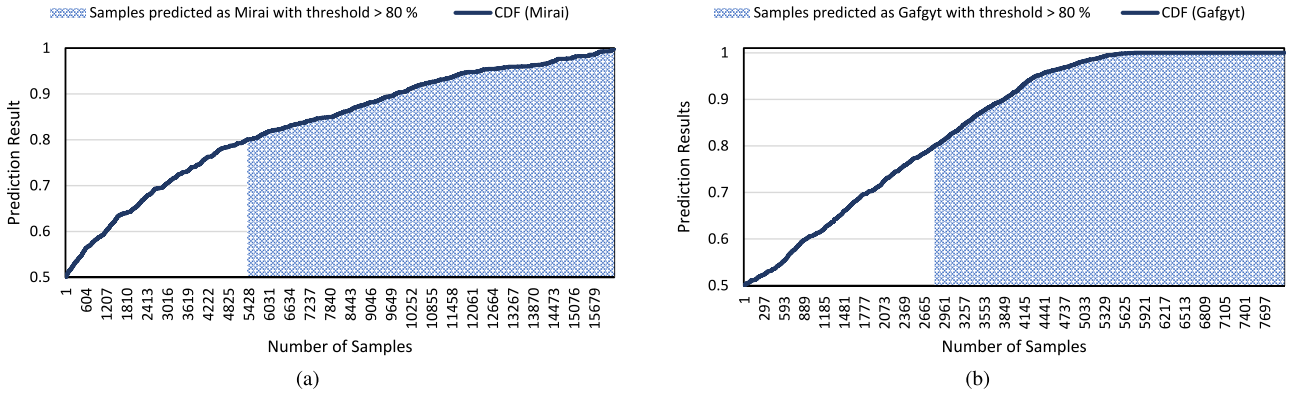


Fig. 6. Cumulative distribution function of the samples predicted as (a) Mirai and (b) Gafgyt.

fact more effective, scalable, usable, and practical, especially when used with a large amount of raw data.

E. Label Prediction for Unknown/Unseen Malware

We leverage the proposed multi-level IoT malware classification model in an attempt to identify the family labels for 24,271 unlabeled (unknown/unseen) malware samples in our dataset. To do this, we trained our multi-level classifier with 10,234 IoT malware samples from four predominant malware families (Mirai, Gafgyt, Tsunami, and Dofloo). Note that despite the lack of ground truth for the unknown/unseen IoT malware, our supervised learning approach will generate multi-label classification outcomes which associate the analyzed samples to non-mutually exclusive malware family labels with a degree of confidence. Accordingly, we assume that a sample with high predicted class value is likely to belong to that specific known IoT malware family. On the other hand, a low class prediction value means that the analyzed sample is less likely to belong to that specific malware family.

The cumulative distribution functions (CDF) for the samples that are predicted as Mirai and Gafgyt are illustrated in Figures 6a and 6b. It is worth noting that the majority of the samples within these two groups are in fact labeled with a relatively high prediction accuracy. Nevertheless, to avoid false positives and achieve a more reasonable/accurate labeling outcome, we consider a sample to be correctly classified as either of these two classes if and only if it was predicted with a threshold greater than 80%. Accordingly, for the first group (Figure 6a), we classify about 67% of the samples (10,786 out of 16,214) as Mirai, while about 55% of the malware samples within the second group (Figure 6b) (4,332 out of 7,923) were classified as Gafgyt.

Moreover, our analysis of the Tsunami-labeled malware samples shows that about 80% of them (108 out 134) were predicted with high confidence (>80%). Interestingly, our classification outcomes did not associate any of the unknown/unseen malware samples with the Dofloo family. This might be due to the fact that our unlabeled dataset was mainly obtained from IoTPOT, which is a specialized IoT honeypot that is designed to interact with Telnet requests (Section V-A), hence, it did not capture Dolfoo attacks since they target other ports/services (e.g., TCP port 2375) [34]. Additionally, it is possible that the Dofloo malware is not actively circulating in the wild due to the specificity of its implementation and targeted vulnerabilities, which have been addressed, respectively. Confirming this however, is beyond the scope of current work and will be considered for future work.

Results Validation: It is important to realize that obtaining the ground truth in terms of malware family labels for the unknown/unseen malware samples is a challenging task since these samples have not been detected or known by major antivirus vendors. To overcome these challenges, we perform a targeted text similarity analysis on the extracted strings from a random sample of IoT malware that were classified by our multi-level model with high confidence (i.e., >80%). We use regular expressions to obtain special textual indicators associated with known malware families while correlating the strings extracted from the unknown malware samples to those known ones through the strings-based similarity analysis. Our assumption is that malware samples from the same family are likely to share string-based indicators that reflect their underlying implementations, functionalities, targeted services, adversarial IP addresses, and command instructions.

Note that the Mirai family consists of the largest number of malware samples in our dataset. Therefore, we validate the outcomes of our classifier by manually inspecting 10,786 Mirai-labeled malware samples and performing text-based similarity analysis on them to associate them with known Mirai samples. To do this, we leveraged known Mirai samples to identify adversarial IP addresses that might be associated with possible C&C servers or targeted victims. Additionally, we identify other possible Mirai indicators such as pre-configured default usernames and passwords that are used during brute-force attacks, commands sequences for communication with the C&C server, Internet scanning/probing instructions and commands, DDoS attack commands, and downloaded malicious payloads/scripts, to name a few.

Our analysis of the Mirai-labeled samples resulted in identifying 3,378 unique IP addresses, among which, about 67% were matching IP addresses extracted from known Mirai samples. While common adversarial IP addresses across different malware samples can associate those samples to the same adversary (e.g., bot master), it might not

TABLE III
SUMMARY OF THE PREVIOUS STATE-OF-THE-ART CLASSIFICATION APPROACHES (NA STANDS FOR NOT AVAILABLE)

	Approach	Feature Type	Classifier	Env.	Dataset (#Samples)	Accuracy
Grayscale Image	Nataraj et al. [**?**]	GIST features	k-NN	Win.	Anubis (9,458)	0.9808
	Ahmadi et al. [**?**]	Local Binary Pattern features	XGBoost	Win.	BIG (10,868)	0.9724
	Gibert et al. [**?**]	128 x 128 Grayscale Image	CNN	Win.	BIG (10,868)	0.9750
	Su et al. [**?**]	64 x 64 Grayscale Image	CNN	IoT	IoTPOT (500)	0.9400
Strings	Tian et al. [**?**]	Printable String Information	Random Forest	Win.	Zoo (1,367)	0.9700
	Nguyen et al. [**?**]	PSI Graphs	CNN	IoT	*NA* (4,002)	0.9240
	Alhanahnah et al. [**?**]	Statistical & String features,	Multi-stage Clustering	IoT	IoTPOT (5,150)	0.9550
Integrated Features	Islam et al. [**?**]	FLF, PSI, API features	SVM, RF, DT, IB1	Win.	*NA* (2,939)	0.9705
	Ahmadi et al. [**?**]	IMG, STR, 1G, ENT, API, etc.	XGboost	Win.	BIG (10,868)	0.9977
	Mays et al. [**?**]	Image & Opcode N-Grams	EL (CNN, NN)*	Win.	BIG (10,868)	0.9724
	Gibert et al. [**?**]	APIs, Bytes & Opcode sequence	Multi-level Deep NN	Win.	BIG (10,868)	0.9975
	This Work	**Grayscale IMG, Malware Strings**	**Multi-level Model**	**IoT**	**IoTPOT (30,000)**	**0.9978**

*Ensemble Learning

necessarily mean that they belong to the same malware family. Therefore, we look for additional Mirai-specific string indicators within the analyzed samples [1], [35]. Interestingly, about 95% of the Mirai-labeled samples contained one or more instances of Mirai-specific strings such as "`POST /cdn-cgi/`", "`/dev/misc/watchdog/`", "`single #`", "`.mdebug.abi32`", "`LCOGQGPTGP`", and "`CFOKLKQVPCVMP`", to name a few. We also found similar commands that are used by Mirai to check for "`wget`" or "`tftp`" installations before using them to downloading and executing further scripts and attack payload from designated servers. Since Mirai is designed to launch DDoS attacks, we also found attack methods such as "`attack_udp_dns`", "`attack_udp_vse`", "`attack_tcp_stomp`", "`attack_tcp_syn`", and "`attack_tcp_ack`", in the analyzed malware strings.

Despite that our validation approach using strings similarities was tested with one IoT malware family (Mirai), our findings shed light on common string/textual indicators that can be used to attribute unknown malware samples to those from known families.These findings can be used to answer our final research question (RQ3), while demonstrating high levels of confidence in the classification outcomes with respect to labeling unseen/unknown IoT malware samples.

Undecisive Labels: In addition to the unknown malware samples that were classified with high confidence, a total of 4,773 samples were labeled with a relatively low prediction values ($\leq$70%), meaning that their labeling is positively indecisive. Since the model can only predict four family labels, an indecisive and low prediction result may indicate that these samples might belong to other known malware families, or are possibly new variants/families, which were not incorporated in training the classifier. Additionally, such results could be due to deployed binary obfuscation and/or possible corrupt files, which hide or scramble the binary contents and lead to possible misclassification outcomes. To overcome these challenges, one can investigate various malware de-obfuscation techniques, while extracting meaningful information that can be used for further similarity analysis in correlation to samples from known malware families.

IV. RELATED WORK

A summary of the reviewed related work along with state-of-the-art ML/DL classification techniques is presented in Table III.

Grayscale Images: An original way to represent an executable file is to reorganize its byte code as a grayscale image, like [11] where every byte was interpreted as one pixel in the image. Considering such malware representation, Ahmadi *et al.* [12] extracted a set of features from the grayscale image, such as Haralick features and Local Binary Pattern features, and achieved an accuracy of 96.90% and 97.24% respectively using the XGBoost classifier. Beppler *et al.* [13] evaluated and compared global (GIST) and local (LBP) descriptors using a multitude of classifiers (e.g., KNN, SVM, DT, RF, CNN). Convolutional Neural networks were also used for classification of malware represented as images. Gibert *et al.* [14] developed a deep learning system based on a CNN that learns visual features from executable files to classify Windows malware into families. Su *et al.* [15] proposed a lightweight solution for detecting and classifying IoT DDoS malware and benign application on IoT devices using a small size convolutional neural network. They achieved 94% accuracy, however their used dataset (500 samples) is very limited in size and diversity, and their considered image size (64x64) is attributed empirically with no consent about the best one. A more prudent resizing of 128x128 has been shown to produce lower variation and maintain a high accuracy rate in all cases [13].

Malware Strings: Extracted malware strings can provide useful indicators associated with a suspect binary and its functionalities. For instance, Tian *et al.* [8] leveraged printable strings extracted from Trojans and viruses to perform classification, and evaluated their approach on a multitude of classifiers (e.g., SVM, RF, Instance Based 1 (IB1), Adaboost), and showed that IB1 and RF classification methods were the most effective (Table III). Alhanahnah *et al.* [9] leveraged N-Gram strings-analysis for correlating and clustering malware samples based on their strings similarities. In addition, Nguyen *et al.* [10] proposed a novel approach for Linux IoT botnet detection based on the combination of Printable String Information (PSI) graph and CNN classifier. Their evaluation

results show that PSI graph CNN classifier achieves an accuracy of 92%. Still, to build PSI Graphs, their approach required generating malware Control Flow Graphs (CFGs), which is a complex task that requires time and domain knowledge. Nevertheless, to the best of our knowledge, no previous work has treated malware strings as a text classification problem and leveraged the use of end-to-end learning and NLP techniques for classification of IoT malware using such information.

Mulitmodal Learning: While these approaches use one representation of the data to extract features that are used for malware classification, in practice, these single-level features might not always be available for analysis (e.g., due to obfuscation). Thus, efforts are being put to design models that leverage multiple data modalities from different malware characteristics. Some approaches like [12] rely on fusing multiple hand-engineered features (e.g., frequency of opcodes, image representation, entropy statistics, etc.) into a single feature vector that is used as input to a traditional ML algorithm. On the other hand, other researchers leverage an ensemble of individual classifiers that process a different modality of data to precisely classify malware [16]–[19]. Alhanahnah *et al.* [9] leveraged the benefits of a multi-level approach for malware clustering and signature generation to detect cross-architecture IoT malware using features such as code statistics feature, high-level structural similarity, and N-gram string features. Gibert *et al.* [17] leveraged the use of multiple features from different modalities of data, combined with deep learning algorithms to detect/classify Windows malware. Despite the fact that their approach produced high classification accuracy, their implemented classifier relies on features (e.g., API function calls, assembly language instructions) that require rather sophisticated reverse-engineering techniques with deep learning network models, which tend to be resource consuming. Yet, in this work, while we deal with certain limitations in the context of IoT, we leverage features that can be retrieved without the need to perform expensive pre-processing and feature-engineering tasks. In addition, while our classification approach produces improved accuracy (99.78%), our implemented DL models can qualify as a lightweight solution for enhancing the security of the IoT paradigm through effective malware classification and threat mitigation.

Transfer Learning: Zhao *et al.* [36] proposed a malware detection method of code texture visualization based on an improved RCNN combining transfer learning, which achieves an accuracy of 92.8%. Bendiab *et al.* [37] proposed an IoT malware traffic analysis approach using deep learning and visual representation for fast detection and classification of new malware. They evaluate their proposed method on a dataset of 1000 pcap files of normal and malware traffic and achieve 94.5% accuracy rate for detection using ResNet50. Both of the above mentioned works [36], [37] rely on deep neural networks with hundreds of layers, such as ResNet50 and ImageNet, whose heavy computational cost of multiple layers can be difficult to handle by resource-constrained IoT devices. In contrary, our multi-level IoT malware classification approach, which achieves an overall accuracy of 99.78%, relies on lightweight CNN and LSTM models that transfer their learned features and weights throughout the proposed architecture, respectively.

Other Features: Recent works have applied deep learning methods on more complex malware representations such as control flow graphs. Yan *et al.* [38] used machine learning techniques to classify malware programs represented as their control flow graphs. Their MAGIC framework achieves high accuracy (99.25%). Nevertheless, their approach is shown to be coarse to detect the malicious programs with a high false negative rate. For instance, some DDoS samples or worms may share the same graph structure as benign software. Our model addresses this by preventing the co-adaptation of the subnetworks to a specific feature type. Therefore, even if two different binaries may share the same characteristics from one modality, our classifier would still achieves good performance by learning distinctive feature from the other data modality, hence resulting in less false negative rate. Alasmary *et al.* [39] proposed an adversarial machine learning detection system for IoT malware based on control flow graph feature representations. Both [38], [39] chose to extract complex and time-consuming features for their analysis, while we leveraged the coupled nature of IoT malware [20], their general lack of obfuscation [20], [21], their lack of diversity [40], [41] and therefore, the ability of deep learning methods to automatically extract a set of descriptive static features from their images and strings without relying on feature-engineering and domain's knowledge. We show that our approach is feasible, accurate and performs well on the used image- and strings-based features in the context of IoT. Yet, Alasmary's approach [39] is robust against Adversarial Examples (AEs) and eliminates the model's vulnerability to AEs. We consider complementing our effective multi-level classifier with an AEs detection component to make it robust against adversarial attacks (see Section V-B).

Despite the recent developments and improvements in the implementation of multi-modal learning techniques, there is still room for improvements, especially in terms of implementing decision-level fusion approaches, where the features learned from different modalities are fused into a single shared representation. In fact, there is little done in literature to successfully leverage the use of deep fusion strategies for IoT malware classification. However, in this paper, we demonstrate the effectiveness of such approaches by implementing and evaluating a multi-dimensional DL classification approach that utilizes empirical data in terms of IoT malware executable binaries, along with strings- and image-based features extracted from static malware analysis, to perform IoT malware classification.

V. Discussion

In this paper, we use next-generation techniques, combining multimodal end-to-end learning and malware features from multiple sources (e.g., static malware analysis), to classify IoT malware executables and attribute them to known families. Yet, while some previous works that rely on transfer learning [36], [37] and a variety of complex features such as control flow graphs [10], [38], [39], API function calls [12], [17], have been shown to be effective in classifying malware samples,

these solutions mainly rely on the ability of domain experts to extract meaningful features. Additionally, these approaches often rely on deep neural networks with hundreds of layers, such as ResNet50 and ImageNet, which are computational heavy and thus, difficult to handle by resource-constrained IoT devices. While this is proven to be an expensive task, we leverage multiple deep learning models to implement a lightweight multimodal learning approach that provides a better or comparable classification accuracy, while enabling scalable and timely classification without any pre-processing or feature engineering.

In addition, we demonstrate the effectiveness of our proposed multi-level model in preventing the co-adaptation of the sub-networks to a specific feature type, therefore making the network less sensitive to the loss of one or more channels of information. Our intuition is that in a real world scenario, it is not possible to rely on one malware characteristic, as some features might be difficult to obtain (e.g., due to obfuscation), or insufficient to differentiate between two malware variants. This is indeed the case with IoT malware, where a significant amount of malware samples in the wild are found to share similar implementations and functionalities due to wide code-reuse, as observed in the case of Mirai and the consequent IoT malware that were based on Mirai's source code [20].

Another advantage of our proposed classification framework is in its modular architecture, which can be complemented by new features from other data modalities to make it adaptable to new contexts (e.g., ransomware). Moreover, such adaptability to various contexts can positively support the generalizability of our approach and the obtained findings. To prove this, further experimentation using various types of malware is required, which can be considered for future work. Moreover, we show the generalizability of our approach to unknown samples and its ability to predict the labels of unseen malware (see Section III-E).

A main outcome of this work is to draw attention to the limitations of existing malware detection and labeling outcomes provided by major antivirus vendors. Specifically, we identify a considerable number of IoT malware samples, which have not been detected or labeled by the antivirus vendors. Consequently, we demonstrate the effectiveness of our proposed IoT malware classification approach to address this limitation, while predicting reliable family labels for such unknown/unseen samples. In addition, our findings show that the majority of those new/unlabelled malware samples were in fact associated with predominant malware families such as Mirai and Gafgyt. These findings highlight a common practice among IoT malware authors, who often rely on reusing/tweaking existing malware implementations to create new malware instances that can be used to target new vulnerabilities in a timely manner.

In line with that, our extended analysis of the new/unknown IoT malware samples unveiled Covid-19 themed Mirai variants, which included different string indicators such as "`/bin/busybox CORONA`" command, or "`Total lockdown is the solution`". More importantly, considering that this work was conducted during the Covid-19 pandemic, the identification of such covid-related IoT malware samples shed light on the rapid evolution of IoT malware, which are designed to abuse global events to their advantage. Therefore, this work contributes to the cybersecurity research by providing means for timely classification of new/unknown IoT malware while attributing them to known families (when applicable).

Finally, it is important to realize that addressing threats associated with the emerging IoT malware is essential for the security of the Internet ecosystem. Moreover, the deployment of next-generation 5G networks and technologies will enable large-scale deployment of IoT devices to support everyday activities of consumers and service providers. Therefore, given the insecurity of the IoT paradigm, the projected increase in the deployed IoT devices will without a doubt amplify the threats associated with IoT malware and IoT-driven cyber attacks. To mitigate such threats in the context of future networks, we envision the integration of our next-generation malware analysis and classification approaches within the intermediate cloud-based monitoring and management systems, to support accurate and real-time malware detection, classification, and attack mitigation, and thus, contributing towards the overall security of the 5G networks.

A. Limitations

A main limitation of this work is to collect a diverse and representatives dataset of a IoT malware samples, which is a challenging task. Indeed, relying on a single source for data might hinder the diversity and generalizability of the obtained IoT malware dataset. Nevertheless, while we leveraged IoTPOT [6] as our main source of data collection, it is worth noting that IoTPOT is considered as one of the most reliable sources of IoT malware data, while providing access to a large number of diverse and recent IoT malware samples. It has also been shown to be more effective than other honeypots (e.g., Honeyd [42]) at capturing various IoT-tailored attacks. In addition, the analysis of Internet-scale scanning activities generated by compromised IoT devices [40], [41], [43] confirms the prevalence of Mirai-like malware attacks/samples in the wild, which are the most dominant variants in the IoTPOT dataset. Additionally, to address the diversity of the data and collect a more representative dataset, we extended our data collection to obtain further IoT malware samples from well-known threat repositories such as VirusTotal and VirusShare. Accordingly, our final dataset contained a representative dataset with more than 70,000 IoT malware samples belonging to four predominant families (Mirai, Gafgyt, Tsunami, and Dofloo).

Another limitation of this work is that we did not consider obfuscated IoT malware samples in the implementation of our proposed malware classification approach. It is important to note that malware authors are inclined to intentionally conceal their identity and therefore use obfuscation techniques to hide data in a malicious executable binary [25]. Therefore, such factors must be considered while building a robust malware classification model, which can also deal with obfuscated samples. Nonetheless, our analysis of the IoT malware samples showed that the majority of them did not employ sophisticated

obfuscation, thus, were de-obfuscated using off-the-shelf tools (e.g., UPX). Therefore, our proposed model is still capable of performing effective classification with respect to the majority of IoT malware samples, while attributing them to known families with high degree of confidence.

B. Future Work

The continuous evolution of malware variants/families might negatively impact the prediction outcomes of deployed ML classification models over time (concept drift). In the context of IoT, the release of the Mirai source code to the public enabled adversaries to reuse the code while creating new Mirai-like malware variants by incorporating new exploits to the existing code [20]. Hence, creating a corpus of malware samples with new versions to share similarities with older versions, respectively. Nevertheless, these similarities are assumed to degrade slowly, while causing the malware population to drift over time. Accordingly, we believe that the prediction quality of malware detectors and classifiers will eventually decay in the future as malware evolution might result in completely new variants [44]. As a result, the aforementioned issue of concept drift has to be taken into account when building a sustainable model for malware detection and classification. For instance, applying similarity measures to track drift in any number of malware families can act as new feature type selection method that will shed light on the feature types that drift the least. Thus, the negligible drift in certain types of features can be exploited in future work by deploying classifiers based on these features thus making them more robust to evolving malware.

Another interesting future research direction is to investigate the problems associated with adversarial ML, where an attacker is assumed to manipulate data and craft adversarial examples using different techniques (e.g., manipulation of static feature) to deceive the detection/classification model [25], [45]. While our proposed framework achieves accurate classification of IoT malware, we believe that future research is required to examine the robustness of our multi-level approach against adversarial ML techniques and detection evasion methods. For instance, improving interpretability can increase our multimodel's robustness to adversarial attacks, by revealing and understanding which features give more weights to the model's performance, and therefore by crafting AEs generated based on these features-level manipulation.

VI. Conclusion

In this paper, we utilized DL architectures to implement and evaluate a novel approach for classifying IoT malware by combining multi-dimensional features extracted from strings- and image-based representations of the executable binaries. Moreover, we addressed the main challenges associated with feature selection/engineering by implementing an end-to-end DL approach that can automatically extract and meaningfully combine features from different representation of the analyzed IoT malware binaries. Our experimental results using 10,234 IoT malware samples from four prominent families demonstrated the effectiveness of our classification approach, with a significantly improved accuracy (99.78%), as compared to conventional single-level approaches. In addition, we demonstrated the capability of the implemented model towards classifying new IoT malware binaries, which were mainly attributed to few known families (e.g., Mirai and Gafgyt). Finally, considering the projected increase in the number of deployed IoT devices, and the pivotal role of these insecure devices in the operation and infrastructure of next-generation 5G networks, this work provides a major step towards the development of practical data-driven tools/techniques for effective IoT threat detection and mitigation, thus, contributing to the security of the IoT ecosystem.

References

[1] M. Antonakakis, "Understanding the Mirai botnet," in *Proc. 26th USENIX Security Symp. (USENIX)*, Vancouver, BC, Canada, 2017, pp. 1093–1110.

[2] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks," in *Proc. IEEE 3rd Int. Conf. Electron. Design (ICED)*, Phuket, Thailand, 2016, pp. 321–326.

[3] *Hajime Botnet Makes a Comeback With Massive Scan for MikroTik Routers*, Radware, Tel Aviv-Yafo, Israel, 2018. [Online]. Available: https://www.radware.com/newsevents/mediacoverage/2018/hajime-botnet-makes-a-comeback-with-massive-scan

[4] J. Vijayan. (2018). *Satori Botnet Malware Now Can Infect Even More IoT Devices*. [Online]. Available: https://www.darkreading.com/vulnerabilities—threats/satori-botnet-malware-now-can-infect-even-more-iot-devices/d/d-id/1330875

[5] L. Pascu. (2018). *78% of Malware Activity in 2018 Driven by IoT Botnets, NOKIA Finds*. [Online]. Available: https://www.bitdefender.com/box/blog/iot-news/78-malware-activity-2018-driven-iot-botnets-nokia-finds/

[6] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the rise of IoT compromises," in *Proc. 9th USENIX Workshop Offensive Technol. (WOOT)*, Washington, DC, USA, 2015, p. 9.

[7] P.-A. Vervier and Y. Shen, "Before toasters rise up: A view into the emerging IoT threat landscape," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2018, pp. 556–576.

[8] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," in *Proc. IEEE 4th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Montreal, QC, Canada, 2009, pp. 23–30.

[9] M. Alhanahnah, Q. Lin, Q. Yan, N. Zhang, and Z. Chen, "Efficient signature generation for classifying cross-architecture IoT malware," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, Beijing, China, 2018, pp. 1–9.

[10] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, "IoT botnet detection approach based on PSI graph and DGCNN classifier," in *Proc. IEEE Int. Conf. Inf. Commun. Signal Process. (ICICSP)*, Singapore, 2018, pp. 118–122.

[11] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Security*, 2011, pp. 1–7.

[12] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proc. 6th ACM Conf. Data Appl. Security Privacy*, 2016, pp. 183–194.

[13] T. Beppler, M. Botacin, F. J. O. Ceschin, L. E. S. Oliveira, and A. Grégio, "L(a)ying in (Test)Bed," in *Proc. Int. Conf. Inf. Security*, 2019, pp. 381–401.

[14] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *J. Comput. Virol. Hacking Techn.*, vol. 15, no. 1, pp. 15–28, 2019.

[15] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 02. Tokyo, Japan, 2018, pp. 664–669.

[16] M. Mays, N. Drabinsky, and S. Brandle, "Feature selection for malware classification," in *Proc. Mod. Artif. Intell. Cogn. Sci. Conf. (MAICS)*, Fort Wayne, IN, USA, 2017, pp. 165–170.

[17] D. Gibert, C. Mateu, and J. Planes, "HYDRA: A multimodal deep learning framework for malware classification," *Comput. Security*, vol. 95, Aug. 2020, Art. no. 101873.

[18] R. Islam, R. Tian, L. M. Batten, and V. Versteeg, "Classification of malware based on integrated static and dynamic features," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 646–656, 2013.

[19] Y. Zhang, Q. Huang, X. Ma, Z. Yang, and J. Jiang, "Using multi-features and ensemble learning method for imbalanced malware classification," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, 2016, pp. 965–973.

[20] E. Cozzi, P.-A. Vervier, M. Dell'Amico, Y. Shen, L. Bilge, and D. Balzarotti, "The tangled genealogy of IoT malware," in *Proc. Annu. Comput. Security Appl. Conf.*, 2020, pp. 1–16.

[21] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, 2018, pp. 161–175.

[22] T. Brosch and M. Morgenstern, *Runtime Packers: The Hidden Problem*, Black Hat USA, San Francisco, CA, USA, 2006.

[23] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AV$_{CLASS}$: A tool for massive malware labeling," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2016, pp. 230–253.

[24] *Virustotal-Free Online Virus, Malware and URL Scanner*, VIRUS TOTAL, Dublin, Ireland, 2012. [Online]. Available: https://www.virustotal.com/en

[25] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proc. Int. Conf. Broadband Wireless Comput. Commun. Appl.*, Fukuoka, Japan, 2010, pp. 297–300.

[26] *FLOSS*, FireEye Labs Obfuscated String Solver, Milpitas, CA, USA, 2016. [Online]. Available: https://github.com/fireeye/flare-floss

[27] M. F. Oberhumer. (2004). *UPX the Ultimate Packer for eXecutables*. [Online]. Available: http://upx.sourceforge.net/

[28] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: Understanding transfer learning for medical imaging," 2019. [Online]. Available: arXiv:1902.07208.

[29] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, 2010, pp. 1–8.

[30] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progr. Artif. Intell.*, vol. 5, no. 4, pp. 221–232, 2016.

[31] (2019). *Autonomio/Talos*. [Online]. Available: http://github.com/autonomio/talos

[32] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018. [Online]. Available: arXiv:1803.08375.

[33] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[34] S. Gatlan. (Jun. 2019). *Exposed Docker APIs Abused by DDoS, Cryptojacking Botnet Malware*. [Online]. Available: https://www.bleepingcomputer.com/news/security/exposed-docker-apis-abused-by-ddos-cryptojacking-botnet-malware/

[35] N. B. Said *et al.* (Nov. 2017). *Detection of Mirai by Syntactic and Semantic Analysis*. [Online]. Available: https://hal.inria.fr/hal-01629040

[36] Y. Zhao, W. Cui, S. Geng, B. Bo, Y. Feng, and W. Zhang, "A malware detection method of code texture visualization based on an improved faster RCNN combining transfer learning," *IEEE Access*, vol. 8, pp. 166630–166641, 2020.

[37] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in *Proc. 6th IEEE Conf. Netw. Softw. (NetSoft)*, Ghent, Belgium, 2020, pp. 444–449.

[38] J. Yan, G. Yan, and D. Jin, "Classifying malware represented as control flow graphs using deep graph convolutional neural network," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, Portland, OR, USA, 2019, pp. 52–63.

[39] H. Alasmary *et al.*, "Soteria: Detecting adversarial examples in control flow graph-based malware classifiers," in *Proc. 40th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Singapore, 2020, pp. 1296–1305.

[40] S. Torabi, E. Bou-Harb, C. Assi, E. B. Karbab, A. Boukhtouta, and M. Debbabi, "Inferring and investigating IoT-generated scanning campaigns targeting a large network telescope," *IEEE Trans. Depend. Secure Comput.*, early access, Mar. 9, 2020, doi: 10.1109/TDSC.2020.2979183.

[41] M. S. Pour *et al.*, "On data-driven curation, learning, and analysis for inferring evolving Internet-of-Things (IoT) botnets in the wild," *Comput. Security*, vol. 91, Apr. 2020, Art. no. 101707.

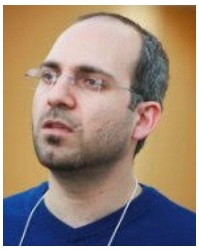
[42] N. Provos, "A virtual honeypot framework," in *Proc. USENIX Security Symp.*, vol. 173, 2004, pp. 1–14.

[43] H. Griffioen and C. Doerr, "Examining mirai's battle over the Internet of Things," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2020, pp. 743–756.

[44] F. Pendlebury, F. Pierazzi, and R. Jordaney, "{TESSERACT}: Eliminating experimental bias in malware classification across space and time," in *Proc. 28th USENIX Security Symp. (USENIX Security)*, 2019, pp. 729–746.

[45] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Security Artif. Intell.*, 2011, pp. 43–58.

Mirabelle Dib (Member, IEEE) received the B.Sc. degree in computer science from the American University of Beirut, Lebanon, in 2019. She is currently pursuing the M.Sc. degree with the Computer Science and Software Engineering Department, Concordia University, Montreal, QC, Canada. Her current research interests are in the areas of IoT security, malware analysis, and machine learning.



Sadegh Torabi (Member, IEEE) received the B.Sc. and M.Sc. degrees (with Distinction) from the Computer Engineering Department, Kuwait University, Kuwait, in 2005 and 2009, respectively, and the M.Sc. degree from the Electrical and Computer Engineering Department, University of British Columbia (UBC), Vancouver, BC, Canada, in 2016. He is currently pursuing the Ph.D. degree with Concordia Institute for Information System Engineering, Montreal, QC, Canada. He was a Research Assistant with UBC from 2011 to 2016. His current research interests are in the areas of cyber security including Internet of Things, cyber-physical systems, and usable security and privacy. He was a recipient of the Concordia University Graduate Fellowship Award from 2016 to 2019.

Elias Bou-Harb (Senior Member, IEEE) received the Ph.D. degree in computer science from Concordia University, Montreal, QC, Canada. He was a Visiting Research Scientist with Carnegie Mellon University, Pittsburgh, PA, USA, from 2015 to 2016. He joined the Department of Computer Science, Florida Atlantic University as an Assistant Professor of cyber security and data analytics in 2016. He is currently an Associate Professor with the Cyber Center for Security and Analytics, Department of Information Systems and Cyber Security, University of Texas at San Antonio. He is also a Research Scientist with the National Cyber Forensic and Training Alliance of Canada. His current research interests are in the areas of operational cyber security, attacks detection and characterization, Internet measurement, cyber security for critical infrastructure, and mobile network security.

Chadi Assi (Fellow, IEEE) received the Ph.D. degree from the City University of New York. In 2003, he joined the Concordia Institute for Information Systems Engineering, Concordia University as an Assistant Professor, where he is currently a Full Professor and holds the University Research Chair. His current research interests are in the areas of network design and optimization, network modeling and network reliability, and smart grids. He was a recipient of the prestigious Mina Rees Dissertation Award for his research on wavelength-division multiplexing optical networks. He is on the editorial board of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON COMMUNICATIONS, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.