Zero-Day Malware Classification Using Deep Features with Support Vector Machines

Rania El-Sayed
Computer Engineering
Cairo University
Scientific Research Group
Cairo, Egypt
raniaes20.eg@gmail.com

Amir El-Ghamry
Faculty of Computers and Information
Mansoura University
Mansoura, Egypt
Scientific Research Group
Cairo, Egypt
amir_nabil@mans.edu.eg

Tarek Gaber
Faculty of Computers & Informatics
Suez Canal University
Ismailia, Egypt
School of Science, Engineering, and Environment
University of Salford, UK
Scientific Research Group
Cairo, Egypt
t.m.a.gaber@salford.ac.uk

Aboul Ella Hassanien
Faculty of Computer and AI
Cairo University
Scientific Research Group
Cairo, Egypt

2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS) | 978-1-6654-4076-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICICIS52592.2021.9694256

***Abstract*—IoT devices are increasingly used every day. However, their limited resources cause them to be vulnerable to any malware, malicious software that causes harm to any device without the user's knowledge. Malwares are called zero-day attacks, a serious threat to internet security since they exploit zero-day vulnerabilities with unknown nature, making them difficult to detect. To solve this problem, the structure of these malware need to be known and analyzed, therefore a small dataset of different types of malware including zero-day attack, is live-captured from network traffic to form 1000 PCAP files representing malware and normal behavior that is used as a source for traffic analysis and malware classification. Most traditional malware detection systems proposed in the literature use signature-based methods, so these systems cannot detect unknown malware types. This paper aims to introduce novel IoT image-based malware traffic analysis approaches (i.e., anomaly detection) using machine and deep learning techniques that can detect unknown malware. To achieve better analysis, PCAP files are represented as RGB images, then the supervised machine or deep learning algorithm is carefully selected to achieve the best results. In this paper, seven supervised learning algorithms with different s and levels of complexity are tested and compared with analysis to have the best one selected. The seven s are divided into two categories: Two-Layer CNN, Four-Layer CNN, VGG16, and under the category of CNN classifiers and Logistic Regression, Support Vector Machine, and K-Nearest Neighbours under the category of Ordinary Classifiers. Experimental results show that the highest performance reached is 94% with the SVM classifier on MobileNetv2 features due to its fast and stable training with fewer resources compared to the other models.**

***Keywords*—IoT, Malware, Binvis, Deep Learning, ResNet50, MobileNetv2, SVM, VGG16, Visualization**

I. INTRODUCTION

IoT devices are smart devices integrated with sensors to collect specific information to interconnect and exchange information with other devices through the internet. In addition, they are characterized by being small with limited resources. Every year, the Internet of Things (IoT) device market is growing where their number is expected to be 30.9 billion units by 2025 according to Statista [1]. Some examples of IoT devices are smartphones, smartwatches, smart locks, doorbell cameras, and computers.

The big evolution of IoT devices is due to its benefits like interacting between devices and saving a lot of time and money by performing daily human life manual tasks that make human life more comfortable [2]. However, their main challenges are the privacy and security issues where they are vulnerable to any type of malware. The attackers can act as a third party on the network connection between IoT devices. Therefore, they can sniff all sent data.

This data may be personal, private, or sensitive (i.e., patient health care data). Attackers can make very harmful actions with this data. This is due to their limited memory and power that makes it harder to apply security mechanisms like encryption since these systems are resource-intensive. Even if the user secured the device with a strong password, this will not protect the device from internet attacks [3]. Another challenge is that hundreds of millions of new malware are created every day by making obfuscation to the structure of malware, creating zero-day attacks that are difficult to detect. All these issues make malware detection a very challenging problem.

Detection systems are classified into signature-based and anomaly or behavior-based systems [4], [5]. The signature-based systems depend mainly on a specific malware pattern or structure like the concept of the hash function, where each malware has different identification. Therefore, these systems cannot detect new malware since its structure is not known. On the other hand, behavior-based systems depend on the behavior or action of the malware so that many malware types can be categorized into the same family. Therefore, these systems can detect new malware based on their behavior that is similar to a certain malware family. Many malware classification solutions have been proposed in the literature. Still, they need a lot of time, and most of them use signature-based systems, making them cannot detect new malware, including zero-day attacks that represent a serious threat [6]. So this paper will propose an anomaly-based algorithm using machine learning, a novel fast and accurate algorithms for malware classification.

Machine learning performance depends mainly on the input dataset. However, malware dataset collection and labeling require a lot of time to sniff data. As a result, only 1000 PCAP files representing malware and normal behaviors are collected in real-time from network traffic. This dataset is publicly available on Kaggle [7] and has two challenges. The data is considered very small, which represents the main difficulty in traffic analysis. In addition, this dataset is somehow unbalanced. Malware samples are more than normal samples, which makes training a bit more challenging. This also causes the accuracy metric to be insufficient for evaluating the model (i.e., it may not give the actual model accuracy). So, it is better to use other evaluation metrics with accuracy to accurately evaluate the models' performance accurately.

The main contribution of this paper is to compare different anomaly-based machine learning classification algorithms for image-based IoT malware classification. Seven approaches are proposed under two categories: CNN Deep-Learning Classifiers (DLC) for image-based malware classification and Ordinary Classifiers (OC) for feature-based malware classification, where supervised machine learning is applied for the two categories. The contribution steps are as follows:

1) Get the binary visualization of each PCAP file by converting each PCAP binary file into an RGB image. Hence, all approaches are image-based malware classi- fication.

2) For the DLC category: Use the output malware images for training every proposed model in this category.

3) For the OC category: Extract the features from the output malware images and take them as the input to each proposed classifier in this category.

4) Finally, compare the results of each proposed model in terms of performance and complexity.

Experimental results show that the model with the highest accuracy is VGG16, with a value of 96%. However, the SVM classifier on MobileNetv2 features achieved the second higher accuracy of 94% with a much simpler model than VGG. The results show that the SVM model is considered the best algorithm in terms of fast and stable training and less resources required.

The rest of this paper is organized as follows: Section II shows other related work. Section III presents our proposed approaches. Section IV lists conducted experiments and their results. Finally, Section V concludes this paper and discusses future work.

II. RELATED WORK

Image-based malware classification has been introduced as a novel approach in the literature. Souri et al. [8] stated a review of different malware detection approaches using machine learning based on two types of malware detection, signature-based and behavior-based. The authors compared different approaches regarding the classifier, the malware analysis technique, the used dataset, and the accuracy. Ngo et al. [9] compared different malware detection algorithms according to the type of extracted data. They compared the algorithms with respect to the accuracy, the rate of false positives and false negatives, the pre-processing time, and the classification time. Baptista et al. [10] first proposed visualizing intrusion PCAP files as binary RGB images through an online visualization tool called binvis. They mentioned that binvis tool compares a PCAP file byte value with its equivalent color according to the corresponding value in the ASCII table. They used SOINN unsupervised model as a classifier where they achieved an accuracy of 94.1%. Shire et al. [11] used a smaller dataset than Baptista, and they trained this dataset on MobileNet pre-trained supervised model and achieved an accuracy of 91.32%. However, Gueltoum Ben- diab et al. [4] used the classifier and achieved a higher accuracy of 94.5% with the same dataset in [11].

III. PROPOSED APPROACH

This section discusses our proposed approaches for image-based IoT malware classification, shows details about the dataset used, and shows how it is collected.

A. Dataset Collection and Visualization

The dataset used is a recorded packet data sniffed or captured from network traffic, normal or malware traffic, especially from layers two to seven in the OSI model, using packet sniffing tools like Wireshark or tcpdump. Packet sniffing tools are used since captured packets can be presented in a human-readable format. The data is recorded in the form of Packet Capture or PCAP files, where their extensions include TCP/IP and UDP network packets in addition to valuable traffic information that is useful for traffic monitoring and analysis.

Moreover, PCAP files are a valuable resource to use for malicious activity detection, including network intrusions. The dataset is publicly available in [7] with a collection of 1000 PCAP files represent normal and malware behaviors. As mentioned before, the dataset is very small and imbalanced. To get a better malware analysis, the PCAP files are visualized into RGB images so that the malware structures and patterns can be learned and analyzed easily. There are many methods for image-based data visualization, but the most suitable one for PCAP files is to use an online binary visualization tool called Binvis [12]. This tool converts every PCAP file to an RGB image according to the ASCII table where every byte in the file is assigned a specific color according to its ASCII character reference [4] as follows:

- Null or Spaces: Black and White
- Printable ASCII Bytes: Blue
- Control Bytes: Green
- Extended ASCII Bytes: Red

ASCII printable characters are most of the characters you can find on the keyboard, including digits, letters, punctuation marks, and a few miscellaneous symbols. The control characters are non-printing characters or characters that do not present written symbols like: 0 (null, NUL , \0 , \$\hat{@}$) and 7 (bell, BEL, \a, $\hat{G}$). The extended ASCII characters use 8-bits instead of 7-bits commonly used with ASCII set to give it an additional 128 characters. The final output for every file in the dataset is a 1024*256 RGB image. Fig.1 shows two image samples, malware and normal.

B. CNN Classifiers

This subsection compares different Convolutional Neural Network (CNN) models concerning their complexity. The general of this approach is shown in Fig.2. After having the PCAP files converted to RGB images, these images are used as the dataset for a supervised learning model to be trained and then classify the input data into legitimate or malware. Listed below are the CNN models tested in this approach:

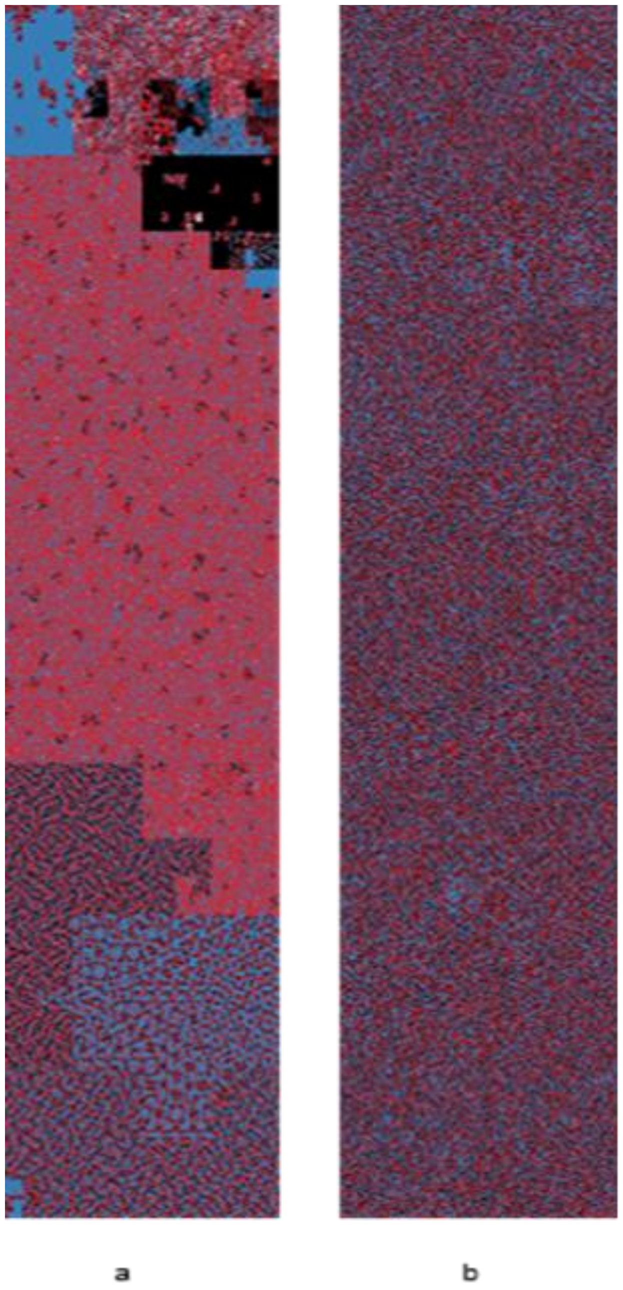

Fig. 1. Image-based Malware Visualization of a) Malware, b) Normal

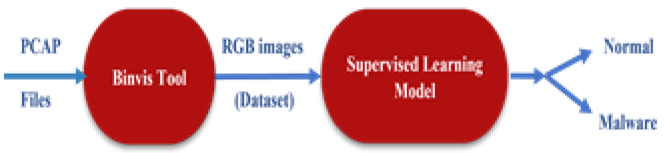

Fig. 2. The Overall Architecture of CNN Classifier.

- Two-Layer CNN: It consists of two convolutional layers followed by a dropout of 20%, fully connected, max-pooling layer, and three fully connected layers then the classifier layer as shown in Fig.3 . Max-pooling layer is added for the sake of dimension and complexity reduction.
- Four-Layer CNN: It includes four convolutional layers followed by max-pooling and batch normalization layers and has two fully connected layers as shown in Fig.4. Batch normalization is used as an optimization technique to enhance model performance and reduce model com- plexity with a more stable training.
- VGG16: It uses VGG16 that consists of 14 convolutional layers and three dense or fully-connected layers [13], pre-trained on ImageNet weights. The classifier layer is replaced with three dense layers to be retrained with new weights before adding the new classifier layer, introducing a dropout layer of 20% before the extended fully connected layers to avoid overfitting. The extended of VGG16 is shown in Fig.5.
- ResNet50: It is a more compex model than VGG16 that uses [14] with 50 layers pre-trained on ImageNet weights. The top layers are replaced with one dense layer to be retrained with new weights before adding the new classifier layer. Fig.6 shows the overall of after adding new layers.

C. Ordinary Classifiers

In this approach, the features are extracted from a pre-trained model on ImageNet weights then these features are introduced to a simple classifier, as shown in Fig.7. [14] and MobileNetv2 [15] pre-trained models are selected for feature extraction. The reason for this is to show the difference when selecting a complex model and another lightweight one. Then, the extracted features from each model are passed to these simple classifiers:

- Logistic Regression (LR): It is a simple supervised machine learning algorithm commonly used for binary classification. It provides a functional form that gives the model the ability to be parametric, semi-parametric, or non-parametric. The classification process is done by calculating the probability of class membership for one category [16]. The probability can be calculated as $P(1|x, \alpha) = \frac{1}{1+e^{-(\alpha.x)}}$ and $P(0|x, \alpha) = 1 - P(1|x, \alpha)$ where the decision boundary separating the classes is all the points x satisfying the equation α.x = 0.
- Support Vector Machine (SVM): It is a supervised learning algorithm commonly used for classification problems for their quick training with large datasets, accurate prediction, and different kernel functions to apply linear classification to non-linear data [17], [18]. SVM is a discriminative classifier that achieves higher classification accuracy than other classifiers by finding the optimal hyper-plane, separating the output classes with the largest margin. The hyperplane is a function $g(x) = \overline{w}\, x + b$ where w is the class, x is the input point, and b is a bias where the distance between the hyperplane and any instance point is $\frac{|g(x)|}{||w||}$ [19]. The most crucial parameter is the kernel function

that maps the data into a higher-dimensional space. Kernel function may be linear, poly, rbf, sigmoid, or precomputed.

- K-Nearest neighbours (KNN): It is a simple, lazy-learning, and non-parametric algorithm that achieves high accuracy with a small number of features. KNN does not build a model. Instead, it directly uses the data for classification. The distances between data points and a randomly initialized point are calculated according to the number of neighbors (k) specified [16].

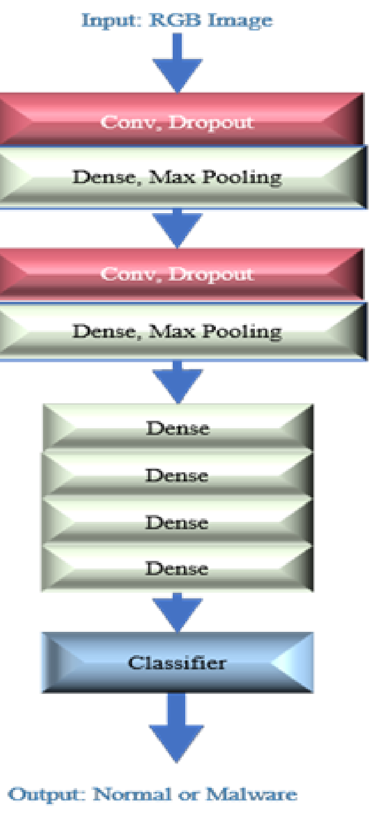

Fig. 3. The overall of Two-Layer CNN

IV. Experiments and Results

This section shows a comparison between all proposed approaches discussed in the previous section in terms of evaluation metrics and complexity. For the experimental setup: all experiments are performed on a Tesla T4 GPU, with 16 GB memory and 320 tensor cores.

As mentioned earlier in this paper, the dataset is small and imbalanced, therefore the accuracy metric is not efficient for evaluating models' performance alone, so the performance metrics in Table.I are used together for a more accurate evaluation of the proposed approaches. The common parameters in all metrics are: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). TP is correctly predicting the positive class, while TN is correctly predicting the negative class. FP is wrongly predicting the positive class, while FN is wrongly predicting the negative class. In the following subsections, the results of each approach are listed.

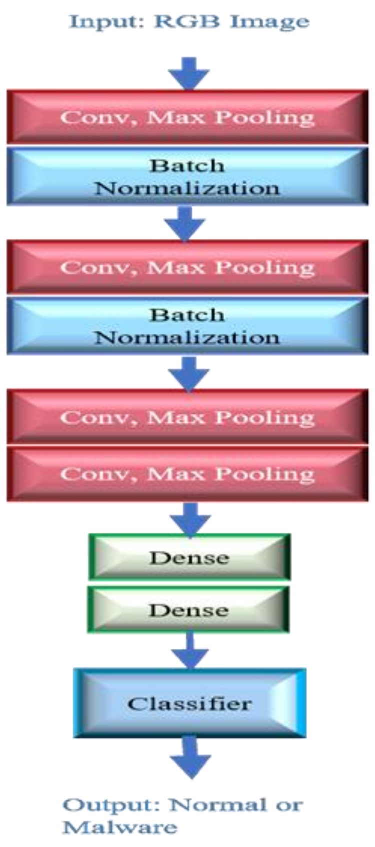

Fig. 4. The overall of Four-Layer CNN

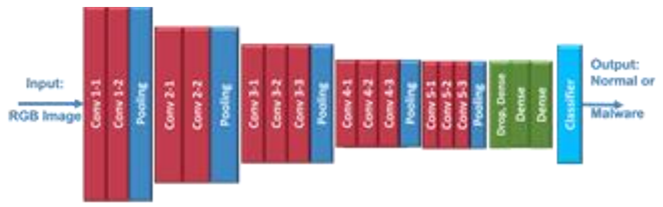

Fig. 5. The overall of VGG16 After Replacing the Top Layers

A. Results of CNN Classifiers

This section lists and compares the training results and evaluates the four CNN models presented in section III-B: Two-Layer CNN, Four-Layer CNN, VGG16, and ResNet50. The comparison is based on the image size (Size), the degree of complexity of each model represented by the number of model parameters (Param), and the prediction accuracy (A). The best model is the model with less image size, complexity, and highest accuracy. Table.II shows that VGG16 achieves the highest accuracy and the second less complexity but the image resolution used is very high, which requires a very long time for processing and training. However, Resnet50 achieves the second-highest accuracy and the least image resolution while being more complex than VGG16 because it has more layers. The Two-Layer CNN has the least accuracy and a middle point concerning complexity. The Four-Layer CNN achieves a relative accuracy with a very simple model compared to and a significantly higher image resolution.

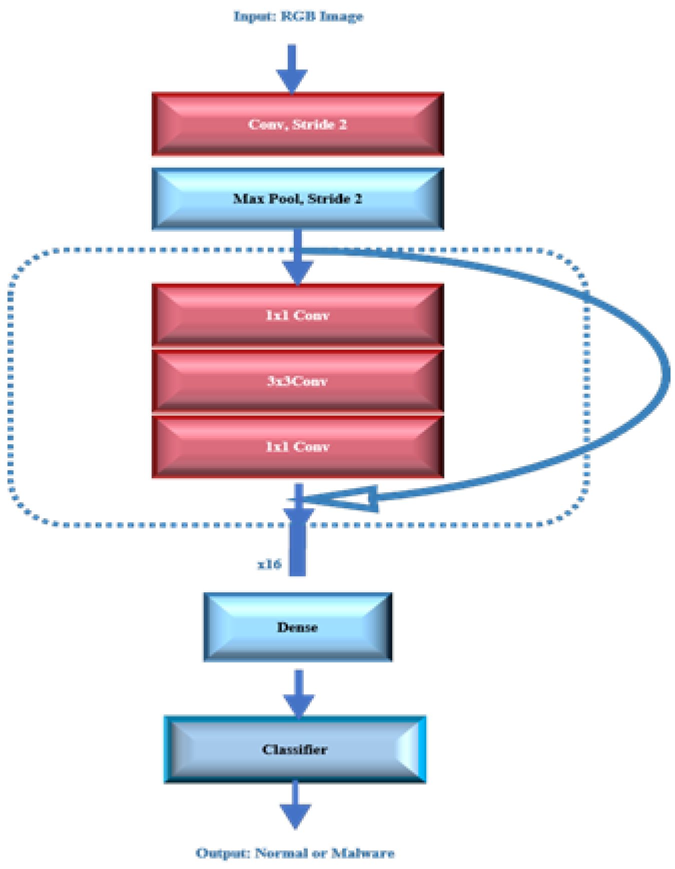

Fig. 6. The overall of ResNet After Replacing the Top Layers

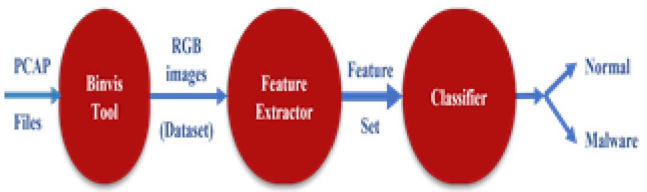

Fig. 7. The Overall Architecture of Ordinary Classifiers

B. *Results of Ordinary Classifiers*

This section compares the feature selection methods mentioned in Section III-C: LR, SVM, and KNN. The comparison is based on these parameters: feature extractor, classifier, and prediction accuracy. All experiments are performed with an image size of 512*512 except KNN, which uses 96*96 image size. The LR model's parameters are set to lbfgs for the solver, 150 for max_iter, and balanced for class weights. SVM kernel function is set to linear and the number of neighbors in KNN is set to K=5. Table.III shows that the SVM classifier achieves the highest accuracy over the MobileNetv2 as a feature extractor. LR classifier achieves a slightly near accuracy compared to SVM even so SVM still performs better concerning accuracy and precision. As mentioned earlier, the KNN classifier performs better with a smaller number of features; however, it cannot achieve better accuracy than 87% with the least number of features. Fig.8 and Fig.9 show the overall results of LR, SVM, and KNN classifiers with and MobileNetv2 feature extractors respectively.

TABLE I
EVALUATION METRICS USED FOR PROPOSED APPROACHES EVALUATION.

Metric, Description and Calculation
Accuracy (A): number of correct predictions per total ($\frac{TP+TN}{TP+FP+TN+FN}$)
Precision (P): predict as positive out of all positive predictions ($\frac{TP}{TP+FP}$)
Recall (R): predict as positive out of all positives) ($\frac{TP}{TP+FN}$)
F1_Score (F1): the mean of precision and recall ($2*\frac{P*R}{P+R}$)

TABLE II
COMPARISON BETWEEN THE RESULTS OF CNN MODELS.

Model	Size	Param	A%	P%	R%	F1%
Two-Layer CNN	512*512	27900K	77	76	76	76
Four-Layer CNN	256*256	76.4K	91	91	90	90
VGG16	1024*256	16700K	96	96	96	96
	224*224	33500K	94	95	92	93

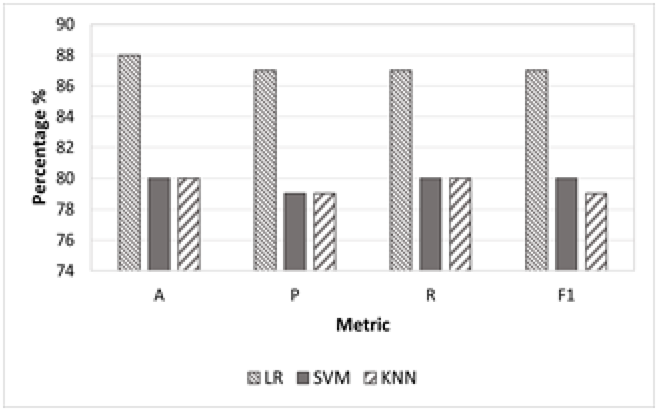

Fig. 8. Overall Results for Ordinary Classifiers With features

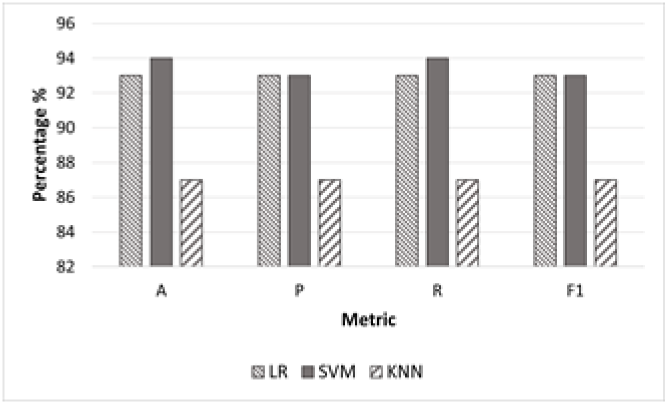

Fig. 9. Overall Results for Ordinary Classifiers With MobileNetv2 features

TABLE III. Comparison Between The Results of Ordinary Classifiers

Metric	ResNet			MobileNet		
	LR	SVM	KNN	LR	SVM	KNN
A	88%	80%	80%	93%	94%	87%
P	87%	79%	79%	93%	93%	87%
R	87%	80%	80%	93%	94%	87%
F1	87%	80%	79%	93%	93%	87%

C. Analysis of the results of CNN and Ordinary Classifiers

From Table.IV and previous experiments, we conclude that CNN models require more processing and resources than Ordinary Classifiers despite the small number of parameters in some CNN models. Table.IV shows the consumed classification time of each classifier. The processing time for the Ordinary Classifiers is calculated as follows: $\mathrm{OC_time} = \mathrm{F_e} + \mathrm{L_d} + \mathrm{C_t}$ where $\mathrm{F_e}$ is the feature extraction time, $\mathrm{L_d}$ is the extracted features load time, and $\mathrm{C_t}$ is the classification time. However, the processing time for the CNN classifiers is calculated as follows: $\mathrm{CC_time} = \mathrm{T_t} + \mathrm{P_t}$ where $\mathrm{T_t}$ is the training time, and $\mathrm{P_t}$ is the prediction time. Although the processing time of the Ordinary Classifiers is close to the simple CNNs and pre-trained models, the latter consumes more processing time and requires intensive resources. The cause of this is that for Ordinary Classifiers, we need to extract the features and load the extracted features only once, though the classification time for each classifier is about two or three minutes maximum. However, CNN and pre-trained models extract the features every time we train the model, though these models require too much time for training all the experiments. In addition, Ordinary Classifiers consume nearly the same processing time if trained on CPU, unlike CNN classifiers which consume more that one hour for training on CPU.

TABLE IV. THE CLASSIFICATION TIME OF ALL CLASSIFIERS

Feature Extractor	Classifier	Time (min)
CNN	Two-Layer CNN	6.40
	Four-Layer CNN	1.70
VGG16	VGG16	8.12
ResNet50		13.7
	LR	5.10
	SVM	7.40
	KNN	7.51
MobileNet	LR	2.00
	SVM	4.10
	KNN	4.20

A comparison between the best results obtained from all experiments based on the model, feature extractor, image size, accuracy, and F1 score is shown in Table.V. Although VGG16 achieves the highest accuracy, it uses very high-resolution images and requires more time for processing and training process. In addition, VGG and ResNet models result in overfitting since the models are too complex for the small dataset used. In our point of view, SVM achieves the best performance compared with other models concerning training stability, fast training, low resources requires, and less complex processing. So, it is more suitable for limited resources of IoT devices.

TABLE V. Comparison Between The Best Results of All Experiments

Model	Feature Extractor	Image Size	A	F1	time (min)
SVM	MobileNetv2	512*512	94%	94%	
		224*224	94%	93%	13.7
VGG16	VGG16	1024*256	96%	96%	8.12

V. Conclusion and Future Work

This paper proposes seven image-based malware classification algorithms using machine learning: Two-Layer CNN, Four-Layer CNN, VGG16, and under the deep learning CNN classifiers and Logistic Regression, Support Vector Machine, and K-Nearest Neighbours under the category of Ordinary Classifiers with feature selection. All algorithms are compared concerning the level of complexity and the output performance where the experimental results show that VGG16 achieves the best with a middle complexity and SVM classifier achieves the second-highest accuracy with a lighter model. In future work, we will explore more advanced optimization techniques such as feature selection.

References

[1] L. S. Vailshery, "IoT device market worldwide," https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/, 2021, [Online; accessed 30-July-2021].

[2] A. Derhab, A. Aldweesh, A. Z. Emam, and F. A. Khan, "Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering," Wireless Communications and Mobile Computing, vol. 2020, 2020.

[3] K. Began, "IoT devices key challenges," https://www.iot-now.com/2020/06/03/103228-5-challenges-still-facing-the-internet-of-things/, 2020, [Online; accessed 30-July-2021].

[4] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "Iot malware network traffic classification using visual representation and deep learning," in 2020 6th IEEE Conference on Network Softwarization (NetSoft). IEEE, 2020, pp. 444–449.

[5] S. Applebaum, T. Gaber, and A. Ahmed, "Signature-based and machine-learning-based web application firewalls: A short survey," Procedia Computer Science, vol. 189, pp. 359–367, 2021.

[6] S. Venkatraman and M. Alazab, "Use of data visualisation for zero-day malware detection," Security and Communication Networks, vol. 2018, 2018.

[7] S. Rana, "MALICIOUS NETWORK TRAFFIC PCAPS-2021," https://www.kaggle.com/sohelranaccselab/trffffffffffffffff, 2021, [Online; accessed 30-July-2021].

[8] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," Human-centric Computing and Information Sciences, vol. 8, no. 1, pp. 1–22, 2018.

[9] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of iot malware and detection methods based on static features," ICT Express, vol. 6, no. 4, pp. 280–286, 2020.

[10] I. Baptista, S. Shiaeles, and N. Kolokotronis, "A novel malware detection system based on machine learning and binary visualization," in 2019 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2019, pp. 1–6.

[11] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, "Malware squid: A novel iot malware traffic analysis framework using convolutional neural network and binary visualisation," in Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Springer, 2019, pp. 65–76.

[12] "Binvis Online visualization Tool," https://binvis.io/#/.

[13] S. Tammina, “Transfer learning using vgg-16 with deep convolutional neural network for classifying images,” International Journal of Scientific and Research Publications (IJSRP), vol. 9, no. 10, pp. 143–150, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[15] H. Pan, Z. Pang, Y. Wang, Y. Wang, and L. Chen, “A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects,” IEEE Access, vol. 8, pp. 119 951–119 960, 2020.

[16] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” Journal of biomedical informatics, vol. 35, no. 5-6, pp. 352–359, 2002.

[17] H. H. Al-Maksousy, M. C. Weigle, and C. Wang, “Nids: Neural network based intrusion detection system,” in 2018 IEEE International Symposium on Technologies for Homeland Security (HST), 2018, pp. 1–6.

[18] V. Apostolidis-Afentoulis and K.-I. Lioufi, “Svm classification with linear and rbf kernels,” July): 0-7. http://www. academia. edu/13811676/SVM Classification with Linear and RBF kernels.[21], 2015.

[19] M. Awad and R. Khanna, “Support vector machines for classification,” in Efficient Learning Machines. Springer, 2015, pp. 39–66.