Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2117883, 9 pages
https://doi.org/10.1155/2022/2117883

Research Article

LiDAR: A Light-Weight Deep Learning-Based Malware Classifier for Edge Devices

Jinsung Kim,[1] **Younghoon Ban**,[1] **Geochang Jeon**,[2] **Young Geun Kim**,[3] **and Haehyun Cho**[2]

[1]*School of Software Convergence, Soongsil University, Seoul 06978, Republic of Korea*
[2]*School of Software, Soongsil University, Seoul 06978, Republic of Korea*
[3]*Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea*

Correspondence should be addressed to Haehyun Cho; haehyun@ssu.ac.kr

Received 17 March 2022; Revised 15 May 2022; Accepted 2 June 2022; Published 14 June 2022

Academic Editor: Xun Shao

With the advent of the 5G network, edge devices and mobile and multimedia applications are used a lot; malware appeared to target edge devices. In the fourth quarter of 2020, 43 million pieces of malware targeting mobile devices occurred. Therefore, a lot of researchers studied various methods to quickly protect users from malware. In particular, they studied detecting malware for achieving the high accuracy with deep learning-based classification models on mobile devices. However, such deep learning-based classifiers consume a lot of resources, and mobile devices have limited hardware resources such as RAM and battery. Therefore, such approaches are difficult to be used in the mobile devices in practice. In this work, we study how a deep learning classifier classifies malware and proposed a novel approach to generate a light-weight classifier that can efficiently and effectively detect malware based on the insight that malware exhibits distinctive features as they are programmed to perform malicious actions such as information leaks. Therefore, by analyzing and extracting distinctive features used by a deep learning classifier from malicious dataset, we generate a light-weight rule-based classifier with high accuracy to efficiently detect malware on edge devices called LiDAR. On an edge device, LiDAR detects malware with 94% accuracy (F1-score) and 85.67% and 328.24% lower usages for CPU and RAM, respectively, than a CNN classifier, and showed the classification time 454.37% faster than the classifier.

1. Introduction

With the introduction of the 5G network, people enter the era of Internet of Things (IoT) in which more devices are connected as developed IoT; edge devices are growing a lot. It is expected that there will be more than 7.49 billion edge device (e.g., smartphones and wearable devices) users worldwide in 2025 [1]. Also, due to the high use of edge devices, multimedia applications are used a lot, and it is seen that cumulative downloads of multimedia applications (e.g., WhatsApp, YouTube, and Facebook) are about 28.4 billion or more [2]. Furthermore, mobile multimedia usage is about 4.23 hours per day which is consumed a lot of time [3]. Unfortunately, due to the severe security threats (e.g., Botnets and man-in-the-middle attack) and major privacy violations (e.g., social security numbers, credit card numbers, and passwords), the use of the edge devices is still risky [4–8]. For example, a single wrong click can launch a malicious program causing damage such as personal information leakage or financial loss. In the fourth quarter of 2020, 43 million pieces of malware targeting mobile devices appeared [9].

Such threats have led to the release of many commercial antivirus products such as Avast, Kaspersky, McAfee, and Norton. However, those antivirus products have a fatal limitation: They cannot detect unknown malware because they mostly rely on the signatures of known malicious applications [10]. To overcome the limitation, a lot

of research works have focused on developing malware detection approaches using deep learning algorithms to protect users [11–23].

Recently, along with the advances in mobile systems-on-a-chip (SoCs), there have been increasing pushes to run malware detection schemes directly on edge devices [11, 12]. This is because executing the schemes on the edge devices can improve the service response time by eliminating the data transfer overhead. It can save up to 46% overhead system consumption than local execution [24]. However, running deep learning-based malware detection approaches on edge devices is still at the nascent stage, since the edge devices are usually energy and resource constrained [25]. Running complex neural networks including many layers, nodes, and many features makes the edge devices consume CPU usage of at least 60% or more (six cores) and RAM usage of about 10 GB [26, 27]. Although previously studied deep learning-based malware detection approaches could achieve very high accuracy, it is hard to apply them on the edge device of which executing resources are limited. Consequently, it is of importance to develop approaches that can employ deep learning-based malware detection on the edge device.

In this work, we propose a novel approach to generate a deep learning-based light-weight classifier, named LiDAR, to enable efficient malware detection at the edge. To build the LiDAR, we first analyze malicious dataset such as SMS spam dataset, e-mail spam dataset, and Android malware dataset. We then extract word tokens from the malware dataset and train a convolutional neural network (CNN) algorithm using the extracted word tokens. Based on the trained CNN algorithm, we extract features that have high weight values using a visual explanation method of decisions from a large class of a CNN-based model, called gradient-weighted class activation map (Grad-CAM) [28], assuming that those features highly contribute to the prediction accuracy. Based on those features, we build a light-weight rule-based classifier.

To show the efficiency and effectiveness of LiDAR, we evaluate it on a workstation as well as the Raspberry Pi. Our evaluation results clearly demonstrate that LiDAR significantly improves the resource utilization as well as the classification time, compared to the state-of-the-art CNN-based classifiers, achieving the feasible accuracy: on average, LiDAR showed 85.67% and 328.24% lower usages for CPU and RAM, respectively, than a CNN classifier, and showed the classification time 454.37% faster than a CNN classifier to detect Android malware, while achieving 93% of the prediction accuracy.

In summary, our contributions are as follows:

(i) First, we analyze general approaches of malware detection process using deep learning-based classification models with spam dataset and Android application dataset

(ii) Second, based on the analysis, we use a deep learning algorithm to find distinctive features of malware. And, we design a light-weight classifier with the high accuracy to efficiently detect malware on edge devices

(iii) Lastly, we thoroughly evaluate a prototype of LiDAR. Also, we compare our classifier against deep learning classifiers to demonstrate the computation resources and classification time of it. Our approach shows better performance of 85.67% and 328.24% lower usages for CPU and RAM than CNN classifiers with 94% accuracy (F1-score)

2. Background and Related Work

In this section, we introduce the advantages and disadvantages of Android malware detection using deep learning-based approaches. We, also, discuss commonly used features of Android malware employed by the previous studies.

2.1. A Limitation of Deep Learning-Based Android Malware Classification Approaches. Recently, a surge of studies were proposed to detect Android malware by using deep learning-based approaches using various features [11–23]. The advance of deep learning algorithms helps achieve the high accuracy by learning distinctive features of data with complex neural networks. Table 1 shows the accuracies (or F1-score) of previous deep learning-based malware detection approaches with algorithms and features used. However, classifiers generated by deep learning algorithms usually require the high computation time and resource usage because many approaches use excessive and detail features based on complex neural networks to achieve the high accuracy [11–19]. Consequently, even though they could achieve the high accuracy, it is difficult to employ them in practical on the most of smart edge devices which have limited computing resources.

2.2. Commonly Used Features for Android Malware Detection. Table 1 summarizes state-of-the-art deep learning-based malware detection approaches. In general, the methods are built based on various features including permissions and/or API calls. Permissions include information on the system-level functionalities, such as current location and network status. API calls are related to the functionalities that an application provides to users (e.g., SMS functions, call functions, and read and write functions). Malicious applications usually exploit specific permissions or API calls, such as reading sensitive data (e.g., a function reading a password) or transferring data (e.g., a function writing to a socket), to leak private data or capture the user behaviors. By using combinations of such features, previous approaches aimed to not only detect malware but also discover its malicious behaviors to assist the wholistic analysis process.However, in edge use cases, it does not necessarily use such detailed features because we merely need to discover whether an application is malicious or not rather than discovering its malicious behaviors in detail. Also, malicious applications usually share distinct features because they are programmed to inflict damages such as sensitive information leaks or financial loss to users. Hence, based on this insight, we propose a way to

Table 1: Summary of deep learning-based malware classification approaches.

Name	Algorithm	Accuracy or F1-score	Features
MalDozer [11]	CNN	96%	API call
DL-Droid [12]	MLP	99%	Permission, etc.
Droid-Sec [13]	DBN	97%	Permission, API call, etc.
Kim et al. [14]	DNN	99%	Permission, component, string, opcode, API
DroidDetector [15]	DBN	97%	API, permission, etc.
DroidDeep [16]	DBN	99%	Permission, API call, action, component, etc.
Li et al. [17]	DNN	97%	Permission, API call, etc.
Ganesh et al. [18]	CNN	93%	Permission
Nix and Zhang [19]	CNN	99%	API call

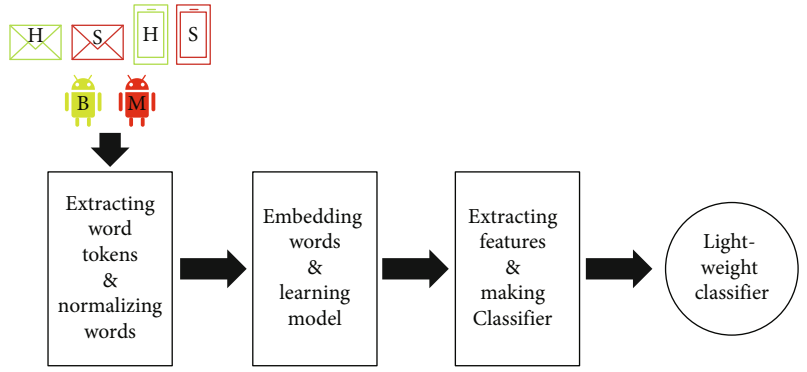

Figure 1: The overview of our approach to generate a deep learning-based light-weight classifier.

Table 2: The summary of our dataset.

	Malicious data				Benign data			
	Spam SMS	Spam e-mail	Android malware		Spam SMS	Spam e-mail	Android malware	
			2019	2020			2019	2020
Training dataset	600	2,953	1,600	1,600	3,857	5,575	1,600	1,600
Test dataset	147	768	400	400	968	1,364	400	400

generate a light-weight classifier that can efficiently detect malicious applications.

3. Overview

We first analyze how deep learning-based classifiers classify malware. Based on the analysis, we aim to design an approach to generate a light-weight classifier with the high accuracy to efficiently detect malware on edge devices. To achieve the goal, we employ a deep learning algorithm to find distinctive features of malware. Since we cannot directly obtain the distinctive features from the trained neural network due to its insufficient explainability, we use Grad-CAM that visualizes how much the features contribute to the classification accuracy. Based on the extracted distinctive features, we build a light-weight rule-based classifier, named LiDAR. It is worth noting that our approach can be applied onto the malware classification problem as well as other types of data which have remarkable features such as scam email. In general, such "malicious" samples in any dataset have distinguishable features from benign samples because attackers create them to have uncommon features shared by benign samples. Therefore, by using distinctive features from malware, we could reduce features and lowering overhead classification for malware detection.

In the following sections, we show how we collected the dataset for this study (in Section 4.1), how we preprocess the dataset (in Section 4.2), how we learn features of the dataset by using a deep learning algorithm (in Section 4.3), how we select important features with a visual explanation technique from the deep learning-based model (in Section 4.4), and how we generate a light-weight classifier based on the features (in Section 4.5).

4. Design

In this section, we demonstrate our approach to generate a light-weight classifier based on the learning result of a deep

Table 3: Malicious and benign features discovered by Grad-CAM.

(a)

SMS spam dataset			E-mail spam dataset		
No.	Weight value	Features	No.	Weight value	Features
1	0.0023	call	1	0.0159	click
2	0.0016	free	2	0.0141	run
3	0.0016	www	3	0.0088	could
4	0.0014	stop	4	0.0086	file
5	0.0013	txt	5	0.0074	remov
6	0.0013	repli	6	0.0070	modem
7	0.0010	cash	7	0.0068	send
	…			…	
800	-0.0017	see	31,860	-0.0144	link
801	-0.0017	heart	31,861	-0.0178	make
802	-0.0018	give	31,862	-0.0237	one
803	-0.0018	weekend	31,863	-0.0334	nbsp
804	-0.0031	get	31,864	-0.0454	emailaddr
805	-0.0038	got	31,865	-0.1133	httpaddr

(b)

Android malware dataset					
2019				2020	
No.	Weight value	Features	No.	Weight value	Features
1	0.0118	android.app-> android.view	1	0.0180	android.app-> android.view
2	0.0094	android.content-> android.content	2	0.0142	android.view-> android.content
3	0.0088	android.content-> android.app	3	0.0135	android.os-> java.lang
4	0.0082	android.webkit-> java.lang	4	0.0130	android.content-> java.lang
5	0.0081	android.app-> android.content	5	0.0124	android.content-> android.app
6	0.0068	android.app->android.os	6	0.0094	android.view-> android.view
7	0.0058	android.view-> android.view	7	0.0092	android.net-> android.net
…				…	
4,667	-0.0051	java.net->java.lang	13,201	-0.0018	android.content.res-> java.lang
4,668	-0.0056	android.view->java.lang	13,202	-0.0023	android.database.sqlite-> android.database.sqlite
4,669	-0.0064	java.io->java.io	13,203	-0.0025	android.webkit-> android.util
4,670	-0.0065	android.widget-> java.lang	13,204	-0.0064	java.io->java.io
4,671	-0.0069	android.content-> android.os	13,205	-0.0065	android.view-> android.util
4,672	-0.0081	android.widget-> android.util	13,206	-0.0068	android.widget-> android.util

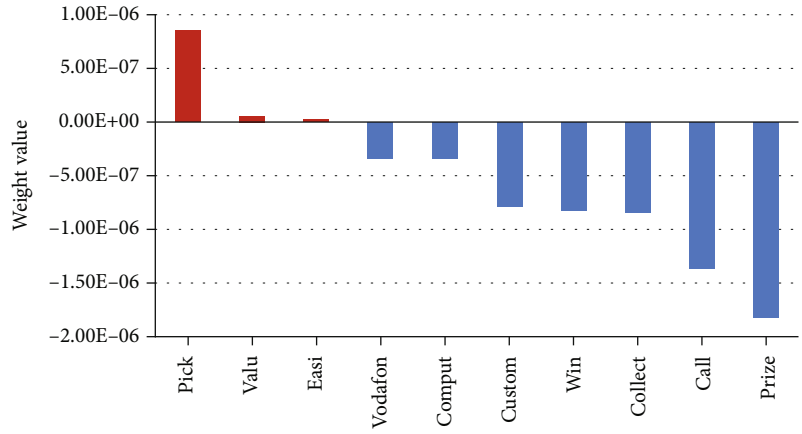

Figure 2: Examples of weights obtained from the SMS spam dataset by using Grad-CAM.

Table 4: The number of word tokens used in our experiments. M: malicious features; B: benign features.

	SMS spam [37]		E-mail spam [38]		Android malware [4] 2019		2020	
CNNc	6,272		82,005		9,211		18,925	
CNNg and LiDAR	M	B	M	B	M	B	M	B
	428	337	12,382	19,483	2,644	2,028	6,576	6,630

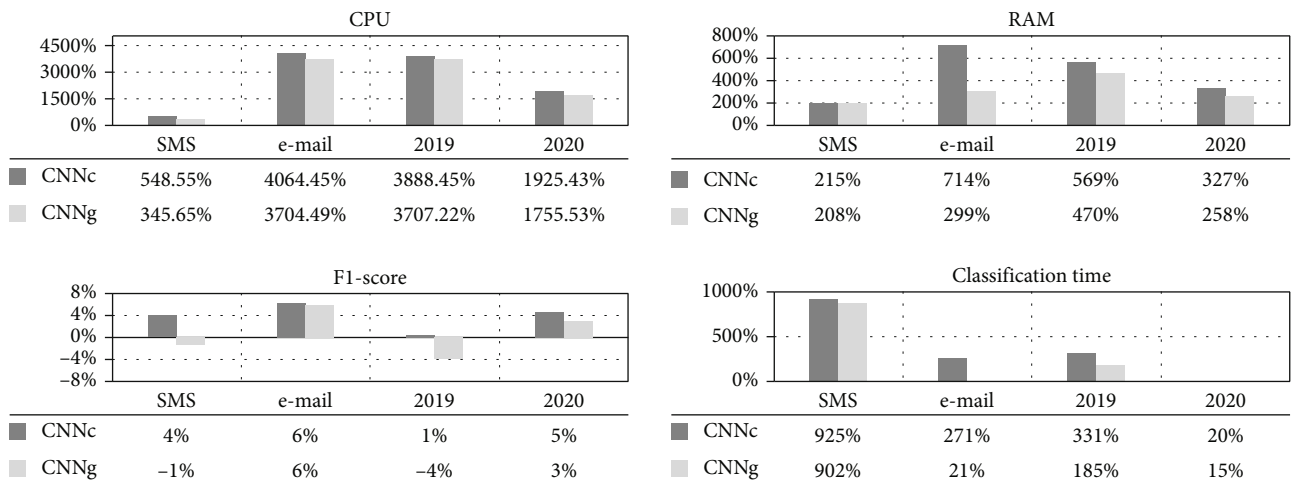

Figure 3: The comparison of F1-scores and the performance overhead of the CNN-based classifiers on the workstation based on the evaluation results of LiDAR.

learning algorithm to classify data samples that have distinctive features such as malware. Figure 1 shows the overview of our approach.

4.1. Dataset. In this work, we collected 24,232 real-world data as in Table 2, which consists of SMS spam message dataset [29], e-mail spam dataset [30], and Android malware dataset appeared from 2019 to 2020 [4]. By using our dataset, we demonstrate that malicious samples of the dataset have notable features to distinguish them from benign data samples, and thus, we can generate a much lighter classifier than deep learning-based models.

4.2. Preprocessing. To make light-weight classifiers, we use word tokens. We, thus, transform the malicious dataset (i.e., SMS spam dataset, e-mail spam dataset, and Android malware dataset) to word tokens. Finally, we remove duplicated word tokens.

4.2.1. Word Normalization. To remove unnecessary texts such as special characters, newline, and stopword for malware classification, we normalize the dataset. We, then, group texts that means the same (e.g., abc@abc.com to email address, https (http) to http address, and $ to dollar). On the other hand, Android malware datasets have many text

Table 5: The evaluation results on the workstation using the three classifiers.

Dataset	Classifier	CPU (%)	RAM (MB)	Classification time (seconds)	F1-score
SMS	CNNc	245.80%	262.13	0.96	0.94
	CNNg	168.90%	256.27	0.94	0.89
	LiDAR	37.90%	83.30	0.09	0.90
E-mail	CNNc	4,451.80%	1,986.491	14.04	0.99
	CNNg	4,067.00%	972.08	4.57	0.98
	LiDAR	106.90%	243.92	3.78	0.93
Malware in 2019	CNNc	3,868.80%	980.04	1.85	0.94
	CNNg	3,693.00%	835.86	1.22	0.90
	LiDAR	97.00%	146.55	0.43	0.94
Malware in 2020	CNNc	3,862.50%	944.33	1.57	0.99
	CNNg	3,538.50%	792.52	1.51	0.97
	LiDAR	190.70%	221.27	1.31	0.94

Table 6: The evaluation results on the Raspberry Pi using the three classifiers.

Dataset	Classifier	CPU (%)	RAM (MB)	Classification time (seconds)	F1-score
SMS	CNNc	176.00%	264.21	3.31	0.94
	CNNg	169.00%	256.54	3.28	0.89
	LiDAR	84.70%	111.01	0.34	0.90
E-mail	CNNc	353.00%	2,029.13	130.73	0.99
	CNNg	345.80%	870.11	45.35	0.98
	LiDAR	167.50%	280.37	18.44	0.93
2019	CNNc	306.70%	582.38	7.51	0.94
	CNNg	294.10%	515.59	5.94	0.90
	LiDAR	189.60%	166.84	1.71	0.94
2020	CNNc	305.40%	638.52	8.15	0.99
	CNNg	303.40%	582.70	6.99	0.97
	LiDAR	172.80%	262.40	6.51	0.94

features (in Section 2.2). Hence, we use Android framework APIs as the main feature of Android malware. We, also, extract API call graphs (ACG) by which we can track data flows between a point where sensitive data is read and another point where the sensitive data is exported by using FlowDroid [31].

4.2.2. Word Encoding for the Malware Dataset. To learn the malware dataset, we convert a preprocessed each word token in the malware dataset to an integer number for the efficiency. When we meet unknown tokens that could not find in the learning process, we map such word tokens to "Unknownword" token. Lastly, add paddings to make the malware dataset the same length.

4.3. CNN Architecture. We employ a simple CNN for the deep learning algorithm [32, 33]. CNN is widely used to find common features of malware word tokens that are frequently used in actual malware dataset [34]. We use a standard convolutional neural network architecture. The input first goes through an embedding layer and then a one-dimensional convolutional layer (Conv1D) with ReLu activations. The last layer is a dense layer after we flattened data into a vector. The Conv1d is trained by a word using kernel size of 1 to capture a feature of each. We also use the Sigmoid activation function, to further classify binary labels.

4.4. Feature Selection. To investigate how different word token features contribute to the accuracy of a CNN classifier, we use Grad-CAM. Grad-CAM enables one to visualize each feature map layer and understand how the input data of a CNN affect the classification. Also, Grad-CAM can extract weight values without architectural changes or retraining. Grad-CAM exploits the feature maps extracted from the Conv1D layers to identify the impact of the features on the classification results. Grad-CAM sorts the feature maps based on the weight values of any class flowing into the final convolutional layer. As a result, Grad-CAM can extract a heat map of weight values for the word tokens which can be used for the light-weight classification.

Table 3 shows extracted features of the malware dataset using Grad-CAM. Higher values indicate malicious features, while lower values indicate benign features.

4.5. LiDAR. To build the light-weight classifier, we identify important features to classify malware from the malware

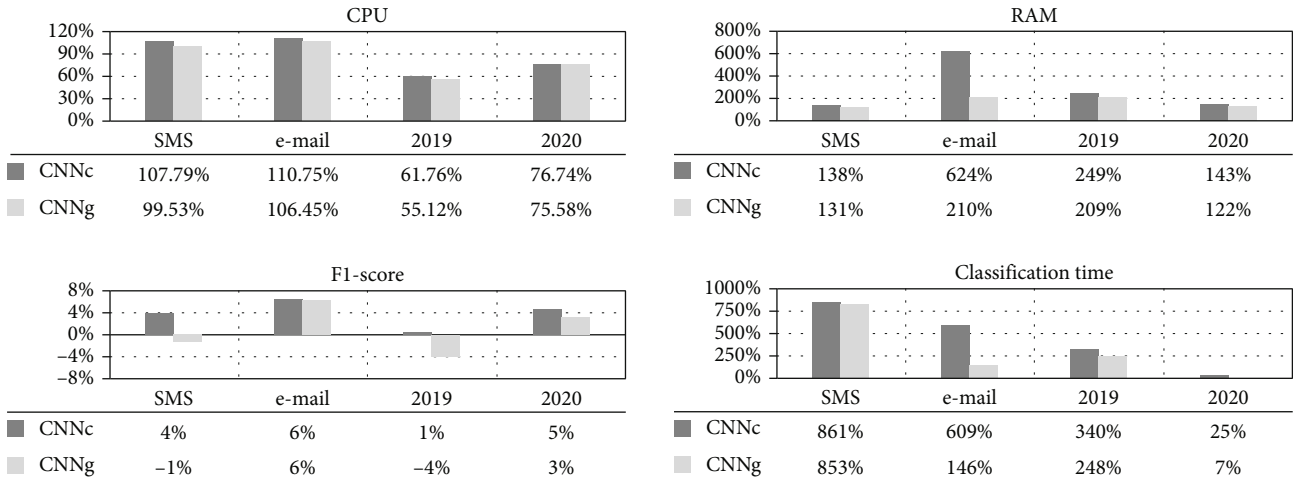

Figure 4: The comparison of F1-scores and the performance overhead of the CNN-based classifiers on the Raspberry Pi based on the evaluation results of LiDAR.

dataset based on the weight values of the extracted features (Section 4.4). As a running example, Figure 2 shows the classified malicious data from the SMS spam dataset based on the weighted values by using the CNN algorithm. In Figure 2, the first three words indicate malicious weighted values, and the others indicate benign weighted values. In this case, an average of more than one-third of the 600 training SMS spam dataset can be identified as the malicious weight values. This means that the malware dataset has more than one-third of distinct malicious features, and the malware dataset can be classified by the number of malicious values. The rule-based classifier can be built based on the observation, by analyzing the number of malicious weight values. Because the CNN classifier does not classify the malware with the context information of SMS spam dataset but with the observed number of distinct words, the rule-based classifier can be built using the following two conditions: (i) When a data has a lot of prelearned words—in this case, we can apply a heuristic condition when a data do not have more than 1/3 of prelearned malicious or benign words. If a data sample has more than one-third of malicious words, we classify it as malware. (ii) On the other hand, if a data sample contains more malicious words than benign words, we classify it as malware. By exploiting distinctive features of malware, we can generate an effective classifier much lighter than a deep learning classifier, albeit we need manual efforts to decide the threshold for classifying malware.

5. Evaluation

In this section, we evaluate our approach to demonstrate its efficiency and effectiveness. We use a Raspberry Pi using the ARM64 architecture as well as a workstation. For the convenience, we refer the CNN classifier to CNNc, CNN classifier using high-weight features to CNNg, and our approach to Light-weight Deep Learning-based Malware Classifier (LiDAR).

5.1. Experiment Setup. We performed our evaluations on a workstation running Ubuntu 18.04 with 20-core Intel Xeon Gold 6230 two CPUs at 2.10 GHz, 256 GB RAM, and a NVIDIA GeForce RTX 2080 GPU. And we conduct experiments on a Raspberry Pi 4 Model B (Rev 1.4) running Ubuntu 18.04 with a 4-core Cortex-A72 (ARM v8), 4GB RAM. We implemented LiDAR by using Python v3.7.1, TensorFlow GPU v1.14.0, Keras v2.2.4, CUDA v11.2, and FlowDroid v1.5 for extracting ACG.

Table 4 shows that the number of words used for performance comparison in each classifier.

5.2. Evaluation Metrics. To explore the effectiveness and efficiency, we used the following metrics.

(1) *CPU Usage*. We consider the maximum workload that a single CPU can handle is 100%, and we show the classifier's CPU usage based on it (e.g., if CPU usage is 200%, it means we need two cores fully to perform a classification)

(2) *RAM Usage*. We measure the resident set size (RSS) of a classifier when it runs

(3) *Classification Time*. We measure the total execution time of a classifier

(4) *F1-Score*. We use the F1-score of classification results to show the effectiveness of each classifier

5.3. Evaluation Results on the Workstation. In this section, we evaluate classifiers on a workstation using malware dataset (SMS spam dataset, e-mail spam dataset, Android malware dataset).

Figure 3 and Table 5 show the experimental results. CNNc used an average of 3,107% of the CPU usage, and CNNg used an average of 2,867%. On the other hand, LiDAR showed an average of 108% of the CPU usage, which is much lower than the CPU usage of CNNc and CNNg. In addition, the RAM usage of LiDAR is also averagely 500.4%

and 311.02% lower than that of CNNc and CNNg, respectively, as shown in Table 4. These results yielded the significant improvement of classification time of LiDAR (averagely 228% and 46.78% faster than CNNc and CNNg, respectively). Nevertheless, LiDAR achieves almost similar F1-score with CNNs and CNNg; the accuracy difference of CNNc and CNNg is only 3.87%. These results imply that LiDAR strikes a good trade-off point between the performance and prediction accuracy.

5.4. Evaluation Results on the Raspberry Pi. Table 6 and Figure 4 illustrate evaluation results of each classifier on the Raspberry Pi. CNNc and CNNg used 285% and 278% CPU usages on average, but the CPU usage of LiDAR is 154% on average, which is 80.98% and 85.67% lower than the CPU usage of CNNc and CNNg, while the RAM usage of CNNc and CNNg is 328.24% and 171.13% on average, which is much higher than that of LiDAR. As a result, LiDAR has an average classification time of 454.37% and 127.95% faster than CNNc and CNNg. Despite the improvement of these results, there is only a small difference in F1-score of 3.87% with CNNs and CNNg, such as the experimental results on a workstation. Consequently, we can observe that LiDAR offers a good compromise between the performance and classification accuracy in any environment.

6. Conclusion

With the advent of the 5G network, a lot of malware targeting IoT devices occurred. Accordingly, a lot of research is on deep learning-based approaches to quickly protect users from malware. However, such deep learning-based approaches consume a lot of resources. In this work, to enable efficient malware detection on the edge devices, we proposed a novel approach to generate a light-weight classifier, LiDAR. We analyzed the SPAM and malware features by using deep learning-based Grad-CAM. Based on distinct features extracted by Grad-CAM, we built LiDAR with a rule-based classifier. Our evaluation results show that LiDAR can effectively detect malware achieving 92.78% of prediction accuracy, while only exhibiting 154% and 205.15 MB of CPU and memory resources, respectively, which resulted in the significant improvement in the classification time: roughly two times faster than a CNN-based deep learning model on average.

6.1. Limitations and Future Works. First off, LiDAR has the out of vocabulary problem as the other deep learning-based approaches have. If our classifier meets an unknown word token, the token is simply ignored. Therefore, to use LiDAR in practice, it is important to continuously learn emerging malware. In addition, similar to the other malware classification approaches, LiDAR cannot detect heavily obfuscated malware because we cannot find effective word tokens from malware if obfuscation techniques such as the class encryption are applied on the malware. We note that classifying unknown and obfuscated malware is a challenging problem, and the limitation is common in deep learning-based approaches. We leave these limitations as future work.

Data Availability

The data used to support the findings of this study were supplied by Jinsung Kim under license and so cannot be made freely available. Requests for access to these data should be made to Jinsung Kim (okokabv@soongsil.ac.kr).

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) Grant through the Korean Government (MSIT) under Grant NRF-2021R1A4A1029650.

References

[1] L. S. Vailshery, "Number of connected wearable devices worldwide by region from 2015 to 2022," 2021, https://www.statista.com/statistics/490231/wearable-devices-worldwide-by-region.

[2] L. Ceci, "Most downloaded mobile apps worldwide from 1st quarter 2014 to 3rd quarter 2021," 2021, https://www.statista.com/statistics/1280313/downloads-top-apps-worldwide/.

[3] L. Ceci, "Time spent per day with mobile non-voice media in the United States from 2019 to 2023," 2022, https://www.statista.com/statistics/469983/time-spent-mobile-media-type-usa/.

[4] VirusShare, "Android malicious applications dataset," 2021, https://virusshare.com/.

[5] K. W. Ching and M. M. Singh, "Wearable technology devices security and privacy vulnerability analysis," *International Journal of Network Security & Its Applications*, vol. 8, no. 3, pp. 19–30, 2016.

[6] A. D. Raju, I. Y. Abualhaol, R. S. Giagone, Y. Zhou, and S. Huang, "A survey on cross-architectural IoT malware threat hunting," *IEEE Access*, vol. 9, pp. 91686–91709, 2021.

[7] M. Al-Hawawreh, F. den Hartog, and E. Sitnikova, "Targeted ransomware: a new cyber threat to edge system of brownfield industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7137–7151, 2019.

[8] H. Haddadpajouh, A. Mohtadi, A. Dehghantanaha, H. Karimipour, X. Lin, and K.-K. R. Choo, "A multikernel and metaheuristic feature selection approach for IoT malware threat hunting in the edge layer," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4540–4547, 2020.

[9] McAfee, "Labs Mobile Threat Report," 2021, https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf.

[10] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Information Sciences*, vol. 231, pp. 64–82, 2013.

[11] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48–S59, 2018.

[12] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2020.

[13] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," *ACM conference on SIGCOMM*, p. 2014, 2014.

[14] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[15] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[16] X. Su, D. Zhang, W. Li, and K. Zhao, "A deep learning approach to android malware feature learning and detection," *IEEE TrustCom-BigDataSE-ISPA*, p. 2016, 2016.

[17] D. Li, Z. Wang, and Y. Xue, "Fine-grained android malware detection based on deep learning," *IEEE Conference on Communications and Network Security (CNS)*, 2018, pp. 1-2, Beijing, China, 2018.

[18] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park, and H. Jeon, "CNN-based android malware detection," in *International Conference on Software Security and Assurance (ICSSA)*, pp. 60–65, Altoona, PA, USA, 2017.

[19] R. Nix and J. Zhang, "Classification of android apps and malware using deep neural networks," in *International joint conference on neural networks (IJCNN)*, pp. 1871–1878, Anchorage, AK, USA, 2017.

[20] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "PICAndro: packet inspection-based android malware detection," *Journal of Internet Services and Information Security (JISIS)*, vol. 2021, no. 2, pp. 1–11, 2021.

[21] J. Jung, H. Kim, S. Cho, S. Han, and K. Suh, "Efficient android malware detection using API rank and machine learning," *Journal of Internet Services and Information Security (JISIS)*, vol. 9, no. 1, pp. 48–59, 2019.

[22] A. L. Marra, F. Martinelli, F. Mercaldo, A. Saracino, and M. Sheikhalishahi, "D-BRIDEMAID: a distributed framework for collaborative and dynamic analysis of android malware," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 3, pp. 1–28, 2020.

[23] R. Casolare, C. De Dominicis, G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Dynamic mobile malware detection through system call-based image representation," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 12, no. 1, pp. 44–63, 2021.

[24] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2410–2422, 2020.

[25] S. Wang, A. Pathania, and T. Mitra, "Neural network inference on mobile SoCs," *IEEE Design & Test*, vol. 37, no. 5, pp. 50–57, 2020.

[26] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," 2019, http://arxiv.org/abs/1906.02243.

[27] J. Liu, J. Liu, W. Du, and D. Li, "Performance analysis and characterization of training deep learning models on mobile device," in *IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 506–515, Tianjin, China, 2019.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *IEEE international conference on computer vision*, pp. 618–626, Venice, Italy, 2017.

[29] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: new collection and results," *11th ACM symposium on Document engineering*, pp. 259–262, 2011.

[30] The Apache Software Foundation, "SpamAssassin public mail corpus," 2006, https://spamassassin.apache.org/old/publiccorpus.

[31] S. Arzt, S. Rasthofer, C. Fritz et al., "Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," *ACM SIGPLAN Notices*, vol. 49, no. 6, pp. 259–269, 2014.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] Y. Kim, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[34] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, "Adversarial examples for CNN-based malware detectors," *IEEE Access*, vol. 7, pp. 54360–54371, 2019.