An Enhancement for Image-based Malware Classification Using Machine Learning with Low Dimension Normalized Input Images

Tran The Son[1]*, Chando Lee[2**], Hoa Le-Minh[3], Nauman Aslam[3], Vuong Cong Dat[1]

[1]*Vietnam – Korea University of Information and Communication Technology, Da Nang, Vietnam*

[2] *National IT Promotion Agency (NIPA), Korea*

[3]*Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Newcastle, UK*

Abstract – This paper proposes a simple and effective model applied for image-based malware classification using machine learning in which malware images (converted from malware binary files) are directly fed into the classifiers, i.e. *k* nearest neighbour (*k*-NN), support vector machine (SVM) and convolution neural networks (CNN). The proposed model does not use the normalized fixed-size square images (e.g. 64×64 pixels) or features extracted by image descriptor (e.g. GIST) for training classifiers as existing models do in the literature. Instead, the input images are normalized and horizontally sized down (the width of the image) to a lower dimension of 32×64, 16×64 or even 8×64 than square ones (e.g. 64×64 pixels) to reduce the complexity and training time of the model. It is based on the fact that the texture of the malware image is mainly vertically distributed as analysed in this paper. This finding is significant for training those devices which have limited computational resources such as IoT devices. The experiment was conducted on the Malimg, Malheur datasets which contains 9339 (25 malware families) and 3133 variant samples (24 malware families) using *k*-NN, SVM and CNN classifiers. The achieved results show that it is possible to reduce the dimension of the input images (i.e. 32×64, 16×64 or even 8×64) while still retaining the accuracy of classification as the same as the accuracy obtained by classifier feeding by the fixed-size square image (i.e. 64×64 pixels). As a result, training time of the propose model reduces by a half, a quarter, and one-eighth compared to training time taken by the same machine learning-based classifier (i.e. *k*-NN, SVM and CNN) feeding by fixed-sized square images, i.e. 64×64, respectively.

Keywords – Image-based Malware Classification; *k*-NN; SVM; CNN; GIST descriptor

1 INTRODUCTION

To detect and classify malware, an anti-virus program is based on two commonly used techniques which are signature-based and behaviour-based [1] detection. The signatures of the malware are usually obtained from known malware by doing static analysis (without execution of the malware). A database built on signatures collected from various malicious objects is applied for malware detection and classification. Though the signature-based detection method is known as very fast and precise, it can be easily bypassed by applying obfuscation techniques (such as encryption, packing, polymorphism, and metamorphism) to generate a new variant [1] [2]. Moreover, the signatures' database (usually based on static analysis) is often manually generated and updated, thus it is time and labour consuming.

Unlike signature-based detection, in behaviour-based detection, behaviours of the malware are acquired and logged during the execution of a given malicious code [1] under a so-called dynamic analysis. By doing so, this approach can detect polymorphic and metamorphic viruses based on their behaviours. However, storing run-time behavioural patterns is considered resource intensive. Furthermore, the behaviour database needs to be updated when a new malware family is found.

Recently, a new approach has attracted a lot of attention for malware classification which relies on image processing [3] [4] [5] [6] [7]. This approach allows a classifier to detect and classify the existence of a malware instance based on the texture of the malware image which is converted from the collected malware binary [6] [7] [8]. This approach does not use signatures or behaviours of malware obtained by static or dynamic analysis as the traditional malware classifiers do, thus overcoming some of weakness of traditional approaches, i.e. signature-based, behaviour-based, as discussed. In fact, it is observed that there is a small change among variants of a malware family, e.g. insertion a set of Non-operation (NOP) instructions into the binary code of a malware instance. Having produced a small change, it induces a big

* T. T. Son is the corresponding author with email ttson@vku.udn.vn; ** Chando Lee was with the NIPA

change in the signatures among variants (i.e. different hash codes, different header sizes, and so on). It is a big challenge for the signature-based malware classification. Meanwhile, a small change in the binary code of a malware instance does not produce a big change in the texture of its corresponding malware image. In other words, it is easy for an image-based malware classifier to detect a new variant without doing dynamic analysis to collect malware behaviours.

According to The European Union Agency for Cybersecurity (ENISA) [9] and Kaspersky [10], though the number of new families observed in 2020 decreases compared to that of 2021 (which may be caused by Covid-19 pandemic), it is still very high, which is 13,905 ransomware new variants and 33 new families detected, especially, Emotet – one of the most dangerous malware continued to aggressively expand its market share in 2021.

Therefore, by converting malware binaries to images and applying machine learning for malware classification, new variants can be easily detected [6] [7] [8]. This is because machine learning models such as *k*-NN, SVM or CNN are able to learn malware features (which are also image features), then detect a new variant based on the similarity recognized. In this approach, the outcomes of the static and dynamic analysis as the traditional approaches required are not applicable, instead the features of malware images are needed. In literature, features of malware images are usually extracted by using an image descriptor such as GIST [11], SIFT, SUFT, KAZE [12] and then fed into the machine learning models for classification. Therefore, the dimension of the malware images used in this approach has a significant impact on the performance of the classification. If it is too big, the classifier needs very long time for training because of high computational complexity. If it is too small, the classifier obtains insufficient information for learning.

Recently, Internet of things (IoT) devices' malware are reported to rapidly increase (according to report of Kaspersky [13] [14] in 2019, 2020). Malware could take control IoT devices and cause a severe attack over the computer networks (e.g. DDoS) [15] [16]. Those IoT devices (home appliances, network cameras, sensors and controllers, etc.) are known as limited computational resources [17]. Therefore, malware tends to decrease its size to easily infect into those devices, also malware classification models applied to those devices should be light-weight or cloud-based (such as the model proposed in [15]) to be able to fit into a limited computational resource device. Once compromised, IoT devices (or end nodes) could be controlled or go into the suspended state or disconnected from the Internet. Thus, they have no chance to upload data the cloud-based malware classifier and get results back for detecting and quarantining the malware [15] [18] [19]. In this circumstance, a lightweight malware classification model built and locally operated at IoT end devices is preferable. In this regards, reducing the computational complexity and training time while still ensuring the accuracy of local malware classification is considered as important.

An efficient solution for this issue is to reduce the dimension of input image as proposed in this paper for which input images are normalized into low dimension ones (i.e. shrink down the width) such that the texture of the re-sized images are still retained to be almost the same as the original ones. The low dimension malware images, then, fed into the machine learning models, e.g. *k*-NN, CNN, SVM for malware classification. This helps the employed classifier still be able to classify a malware sample with a high accuracy but significantly reduce training time and computational complexity compared to that of the existing models reported in the literature [3] [4] [6] [7] [8].

The rest of the paper is structured as follows. Section II reviews existing image-based malware classification models, also their advantages and drawbacks. Section III proposes our image-based malware classification using machine learning with low dimension normalized input images. Experiment setups and results are presented in Section IV. Finally, Section V concludes the paper.

2 Related Work

Nataraj *et at.* [3] are known as pioneer researchers for image-based malware classification. In their model, each malware binary is converted into a grayscale image constructing by a sequence of bytes and stored in a 2D matrix (corresponding to the height and the width of an image). Each byte indicates a grayscale pixel which has a value ranging from 0 to 255. The authors in [3] used GIST descriptor to extract the features of the input image applied for training the *k*-NN since GIST can handle a huge dataset of images better than other descriptors [20]. Their experiment conducted on Malimg dataset containing 25 different

malware families and achieved a high accuracy of almost 98%. However, using GIST or any other descriptors is considered as complicated and time-consuming [21]. That also did motivate the author in [21] to combine two commonly used feature extraction algorithms consisting of the best subset selection and forward stepwise selection to reduce feature extraction time.

In another work, the authors in [4] used the feature extraction tool published in [22] to get the features needed for their image-based malware classification model using *k*-NN and deep learning. This model was pre-trained on the ImageNet dataset which contains 1.2 million malware images in 1,000 classes to improve the classification performance. However, their experiment results obtained the accuracy of 92% approx. for classification using deep learning, which is not very high accuracy compared to the existing works in the literature.

In order to improve the performance of the image-based malware classification using deep learning, S. Yajamanam *et al.* [5] have just selected 60 highest ranking features among 320 ones extracted by GIST and then fed into their classifier. However, their proposed model achieved only 92% accuracy. In this work, the authors also stated that it is possible to apply deep learning to raw images without using GIST for extracting features but they did not show how to implement it and achieve results.

In an effort *not* to employ feature engineering, Quan Le *et al.* [6] applied deep learning for raw input images. To do so, raw input images are converted to 1D fixed-size vectors by applying a generic image scaling algorithm prior to feeding into the deep learning model, i.e. CNN. The accuracy of their classification model achieved 95~98% conducted on various datasets. However, this proposed model takes long time for running the generic image scaling algorithm to convert images into 1D vector (i.e. 191.2 seconds).

In another work, Z. Cui *et al.* [8] proposed an image-based malware classification model using deep learning to solve the imbalance problem of a dataset. In their experiment, raw images in the Malimg dataset [3] are converted to 2D vector and fed into the CNN after balancing them by their own proposed algorithm. To facilitate the CNN for classification, all input images are re-sized to be the fixed square images. As a result, the accuracy of their proposed model achieved 94.5%.

Using the same approach, the authors [23] proposed the Extreme Learning Machine (ELM) model which is one of simple machine learning architectures besides using CNN for image-based malware classification. Their experiment was conducted on different situations (i.e. image sizes, convolutional layers, and filters) show that the performance (i.e. accuracy) of the ELM and the CNN are almost the same on the Malimg dataset [3] but training time of the ELM is much faster than that of the CNN due to its simplicity.

A recent study done by authors in [7] proposed a hybrid model for image-based malware classification to detect known and unknown malware using machine learning. It composed of three different subsystems designed based on supervised and unsupervised learning techniques. The accuracy of their proposed model achieved above 98% which is very high compared to that of others.

It is on the fact that all above-mentioned studies require a conversion of malware binaries into images [3] [4] [6] [7] [8] before classifying. After conversion, the malware images are observed to be in different dimensions (width × height in pixels) depending on the size of the input malware binaries. Therefore, converted malware images need to be normalized to the same size so that a classifier can learn and recognize the similarities among images, thus classifying an input malware sample. In the literature, malware images are usually normalized to 64×64 pixels for training a machine learning-based classifier [3] [4] [6] [7] [8] even for those which are very small size such as IoT malware as discussed above [15].

In terms of image processing, a pixel is represented by one byte in a greyscale image [3], hence 64×64 pixels are represented by 64×64 bytes in a computer. If there is an input malware image which has a smaller size than 64×64 bytes (or 4KB), it is required to insert additional bytes for normalization. It results in a reduction of accuracy for image-based malware classification due to extra information inserted. For example, in [15] the accuracy is obtained as lower than 80% for 3-class classification.

This study has found that the texture of a malware image is vertically distributed along the *y*-axis (i.e. the height of the image) rather than *x*-axis (i.e. the width of the image). In other words, the width of the malware image is of little significance compared to the height of a malware image (see analysis in Section 3.2). Therefore, the width of the input malware image is able to be re-sized down without too much impacting on its texture, thus little affecting the accuracy of malware classification. Motivated by this

finding, the next section proposes an image-based malware classification model using machine learning with sized-down input images to improve classification performance (i.e. computational complexity and training time).

3 Malware Image and Texture Analysis

3.1 Malware Image

It is well-known that a malware instance usually appears in the portable executables (PE) format which is actually a binary file (so-called malware binary). A malware image is an image converted from malware binary. In order to convert a malware binary into an image, this work treats the sequence bytes of the malware binary as the bytes of a grayscale image in a given image format (e.g. PNG, BMP). Each byte represents a grayscale pixel of an image, which has the value of 0 to 255. Thus, a 1-D image vector is generated as follows

$$I = \{b_1, b_2, ..., b_N\} \quad (1)$$

where b_i is the decimal value of the i^{th} byte; N is the total number of bytes generated.

Because the malware binaries have different sizes (e.g. 4KB, 10KB, 100KB, etc.), their corresponding malware images are also in different sizes (width × height). Therefore, a malware image I can also be represented as a 2-D matrix to express the width and the height of an image [3]. In fact, the width of an image is usually assigned by given value such as 32, 64, 128 pixels. The height, hence, is variable depending on the size of the malware binary. TABLE 1 shows the various recommended widths of the malware image corresponding to the size of its malware binary [8].

TABLE 1. Malware Binary vs. Image Width

Malware binary	Image width (pixel)
<10 kB	32
10 kB ~ 30 kB	64
30 kB ~ 60 kB	128
60 kB ~ 100 kB	256

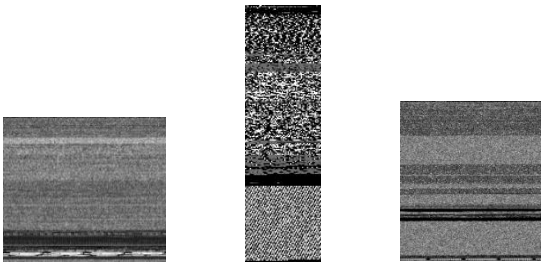
It is noted that the bigger the malware binary, the wider the width of the image. Thus, the width of the malware image (generated by a malware binary) could larger than 256 pixels depending on the size of the malware binary but they are not depicted in TABLE 1 for brevity. Mathematically, the vector I can be treated as a 2-D matrix as follows

$$I = \begin{bmatrix} i_{11}, i_{12}, ... & , i_{1w} \\ i_{21}, i_{22}, ... & , i_{2w} \\ ..... & \\ i_{h1}, i_{h2}, ... & , i_{hw} \end{bmatrix} \quad (2)$$

where i_{11}, i_{12}, i_{21}, ...are determined by sequence bytes of the vector I.

The matrix I is also known as the *grayscale matrix* of a malware image in which each value b_i represents a grayscale pixel of the malware image. The value of 0 indicates the black pixel and 255 indicates the white pixel; and in-between values are different shades from black to white.

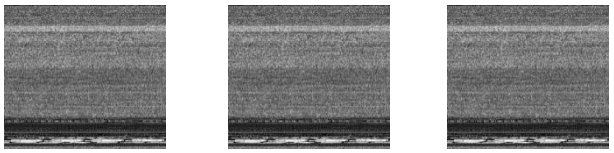
It can be seen that different malware instances appear in different malware binaries (i.e. total number of bytes, header's size, number of sections, etc.). Therefore, they produce different malware images with different sizes and textures as illustrated in Fig. 1 (a), (b), and (c). As the figure shown, three different malware instances, i.e. Autorun.K, Dialplatform.B, Swizzor.gen!E provided by Malimg dataset [3] produce three different malware images.



(a) (b) (c)

Fig. 1. Malware images of (a) Autorun.K; (b) Dialplatform.B; (c) Swizzor.gen!E

Another important finding derived from malware images is that malware images of variants of a certain family have a similar texture as illustrated in Fig. 2 (a), (b), and (c). It means that a small change made on the original malware binary (e.g. inserting some NOP instructions in the binary code) does not induce a big change in the texture of the malware image. It facilitates an image-based malware classifier to detect an obfuscated or new malware variant generated. On the contrary, these small changes cause a big difference in malware signatures or even malware behaviours, thus causing a big challenge for a traditional malware classifier to deal with an obfuscated or new variant generated.



(a) (b) (c)

Fig. 2. Three randomly picked-up variants of the Autorun.K family provided by Malimg dataset

3.2 Texture Analysis

This section provides an analysis on the texture of the malware image and justifies that the texture of a malware image is mainly distributed along vertical direction (i.e the height of the image) rather than horizontal one (i.e. the width of the image). Note that this analysis is *not* a pre-processing step on malware images for the proposed model mentioned in Section 4.

To do so, this paper adopts and applies the most commonly used texture measures proposed by R. M. Haralick [24] to analyse the texture of the malware images. Those texture measures are based on the Grey Level Co-occurrence Matrix (GLCM) which is structured from the grayscale level matrix I (see Eq. (2)) by calculating properties of the spatial relationships among nearest-neighbour pixels in four orientations, i.e. 0, $\pi/4$, $\pi/2$ and $3\pi/4$. The GLCM is a two-dimensional matrix in which each element $P(u,v)$ represents the frequency of occurrences of a pair of pixels in a spatial relation determined by the distance d and the angle θ. Let u, v be the grayscale levels, the element $P(u,v)$ can be determined by counting the number of spatial relationships between two pixels as follows

$$P(u,v) = \{I(m,n) = u, \quad I(m+d, n+d) = v\} \text{ for each } \theta \tag{3}$$

where

$m = 0, 1, 2, \ldots, h-1$;
$n = 0, 1, 2, \ldots, w-1$;
$\theta = 0, \pi/4, \pi/2, 3\pi/4$.

Based on GLCM , texture correlation f_{cor} is defined as follows [24]

$$f_{cor} = \frac{\sum_{u=1}^{N_g}\sum_{v=1}^{N_g}(uv)\frac{P(u,v)}{R} - \mu_1\mu_2}{\sigma_1\sigma_2} = \frac{\sum_{u=1}^{N_g}\sum_{v=1}^{N_g}(uv)\,p(u,v) - \mu_1\mu_2}{\sigma_1\sigma_2} \quad (4)$$

where N_g is the number of grayscale levels; R is the total number of pairs of pixels u and v with respect to d and θ; μ_1, μ_2, σ_1, σ_2 are means and standard deviations of marginal distribution associated with the probability $p(u,v)$ in a bi-direction of the spatial relation between two pixels. Since the GLMC is symmetric [24], $\mu_1 = \mu_2$ and $\sigma_1 = \sigma_2$. They are defined as follows

$$\mu_{1,2} = \sum_{u=1}^{N_g} u \sum_{v=1}^{N_g} p(u,v) = \sum_{v=1}^{N_g} v \sum_{u=1}^{N_g} p(u,v) \quad (5)$$

$$\sigma^2_{1,2} = \sum_{u=1}^{N_g} (u-\mu_1)^2 \sum_{v=1}^{N_g} p(u,v) = \sum_{v=1}^{N_g} (v-\mu_2)^2 \sum_{u=1}^{N_g} p(u,v) \quad (6)$$

From Eq. (2), it can be clearly seen that pixels in the rows of the grayscale matrix I (horizontal orientation) have closer spatial distance than those in the columns (vertical orientation). It is because the matrix I is composed of a sequence of bytes provided by the malware binary, i.e. b_1, b_2, b_3... migrating from left to right (row) and top to bottom (column) of the matrix.

According to the First Law of Geography [25], the near pixels are more related than the distance pixels, therefore the $P(u,v)$ of GLCM matrix for horizontal orientation ($\theta = 0$) is greater than $P(u,v)$ of GLCM matrix for vertical orientation ($\theta = \pi/2$)

$$P_{\theta=\pi/2}(u,v) = P_{\theta=0}(u + d_{\max}, v + d_{\max}) < P_{\theta=0}(u,v) \quad (7)$$

where $d_{\max}$ is the width of the image, i.e. $d_{\max} = w$.

Thus, texture correlation in the horizontal direction (along the width of the image, $\theta = 0$) is greater than that of the vertical direction (along the height of the image, $\theta = \pi/2$)

$$f_{cor}(\theta = 0) > f_{cor}(\theta = \pi/2) \quad (8)$$

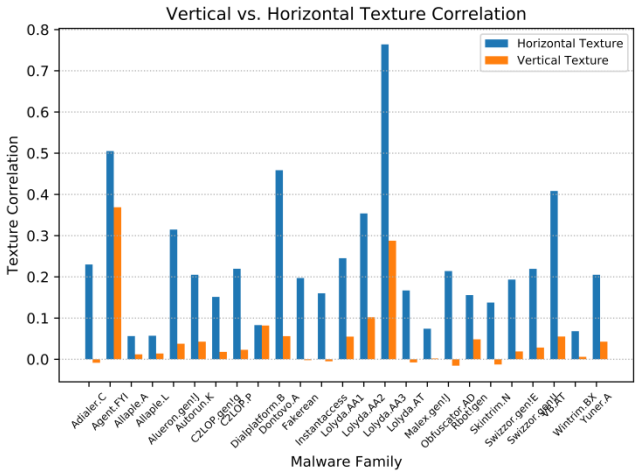


Fig. 3. A comparison between the vertical and horizontal correlation of all malware families on Malimg dataset

Since the texture correlation measures the linear dependency of grey levels on those of neighbouring pixels, Eq. (8) shows that grey levels of pixels distributed along the width of the image ($\theta = 0$) are more dependent and more predictable than that of pixels distributed along the height of the image.
In Fig. 3, it is observed over the Malimg dataset that all the malware families have their horizontal texture correlations much stronger than their vertical ones which means that the horizontal pixels seems to be close (similar) to each other (or consistent) and vertical pixels seem to be far different (or dissimilar) to each other. Therefore, it is easy to recognize the dissimilarity (or sharp breaks) on the vertical direction. In other words, the texture of the malware image is mainly distributed in vertical direction (see examples in Fig. 1, Fig. 2, and Fig. 4).This observation is agreed with the abovementioned analysis.
Fig. 4 shows four versions of a malware image of Dontovo.A malware family in various widths: 64, 32, 16, and 8 pixels in which the 64-pixel-width image is the original one. Though the width of the malware image is compressed down, it is observed that the texture of sized-down malware images keeps almost the same as that of the original one.

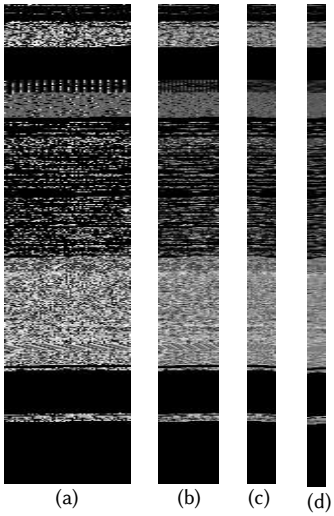


Fig. 4. Four malware images of the *same* instance (Dontovo.A) with the width of (a)64, (b)32, (c)16 and (d)8 pixels

This enables an image-based malware classifier to reduce the dimension (i.e. width) of the input image to improve the classifying performance (i.e. training time) without too much impacting on the accuracy of classification.

To resize an image, a re-sampling technique should be applied to create a new version of the original image in a different size. There are several algorithms applied for re-sampling such as Nearest Neighbour, Bilinear, Lanczos, etc. [26] [27]. In principle, all these algorithms rely on the neighbour pixels of the original image $I(x)$ to interpolate new pixels of the resized image $I'(x)$, where x represents a point sample (that is, a pixel is a point sample). In particular, a grayscale value of the new pixel of the resized image is generally created by replicating or calculating the mean value of neighbours pixels' values. The new image $I'(x)$ is represented as follows

$$I'(x) = I(Kx) \tag{9}$$

where K is the factor of resampling. For example, if the size of the image reduces a half, $K = 2$.
This paper focuses on the down-sampling in the horizontal direction (width) of the malware image by a K factor. This means that width w' of the resized malware image reduces K times compared to that of the original one w. Thus,

$$w' = \frac{w}{K} \tag{10}$$

Then the grayscale matrix $I'(x)$ of the resized image is defined as follows

$$I' = \begin{bmatrix} i'_{11}, i'_{12}, \ldots\ldots & , i'_{1w'} \\ i'_{21}, i'_{22}, \ldots\ldots & , i'_{2w'} \\ \ldots\ldots & \\ i'_{h1}, i'_{h2}, \ldots\ldots & , i'_{hw'} \end{bmatrix} \quad (11)$$

where pixels of I' (i.e. i'_{11}, i'_{12}, ...) are generated by retaining one of the original pixels (e.g. Nearest Neighbour algorithm) or interpolated based on neighbours (e.g. Bilinear algorithm).
Based on Eq. (11), it is said that when the width w of an image reduces to w', i.e. K times, not only the horizontal texture correlation of the resized image increases, but also its vertical texture correlation increases because the distance among pixels reduces.
In other words, the higher the K value is, the stronger the horizontal and vertical texture correlation is obtained since the distance among pixels is closer. It is agreed to the analysis shown in Fig. 5 in which the texture correlation of 8-pixel-width malware image (with respect to vertical distance) is stronger than others, i.e. 16-pixel-width, 32-pixel-width, 64-pixel-width (original one) malware images of the Dontovo.A malware. It is noted that in the negative correlation region, texture correlation of the resized image is also stronger than that of the original one.
More importantly, as depicted in Fig. 5, the variation of texture correlations of the resized images with respect to the vertical distance keeps almost the same as that of the original one (i.e. 64-pixel-width image – the blue curve). This allows us to shrink down the width of the malware image while retaining its texture such that it does not impact much on the accuracy of malware classification.
However, if the width of the malware image is reduced considerably, then the spatial relationship among pixels increases. This reduces the dissimilarity among pixels of an image, thus, generating an unclear texture across the image, therefore impacting on the accuracy of classification. As shown in Fig. 5, when the width of the malware image is reduced from 64 pixels to 32, 16, 8 pixels, the sharp breaks in the vertical direction of the malware image is also reduced.

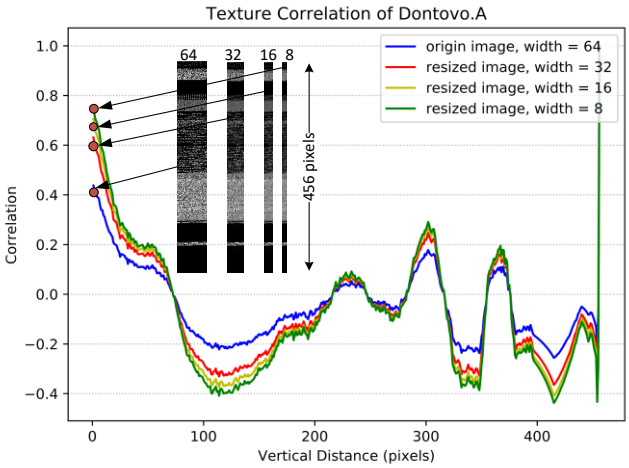


Fig. 5. Texture correlations of four resized versions of the Dontovo.A malware image in different widths

Based on the aforementioned analysis, the next section proposes an image-based malware classification model using machine learning to which input malware images are sized down their width to speed up training process while still remaining the accuracy of classification. This step is carried out in the normalization process, before training the machine learning model as discussed in the next section.

4 The Proposed System Model

This section proposes a model applied for image-based malware classification using machine learning, i.e. CNN with low dimension normalized input images. The key factor of the proposed model is to *normalize and size down* the width of the malware images before training in order to improve the performance of

classification, i.e. reduce training time. This means that the input images of the proposed model are normalized to the same size but they have smaller width (32×64, 16×64, 8×64 pixels) than that of existing models (i.e. fixed-square of 64×64 pixels). The system model is as follows

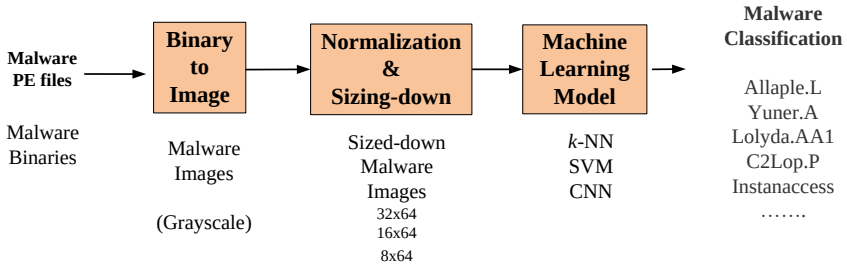


Fig. 6. The proposed system model

As Fig. 6 shown, the malware images are converted from malware binaries (PE files) as discussed in Section III-A, then normalized and sized down the width before feeding to the CNN for classification.

The achieved results of the proposed model are compared to that of the previous existing models using features extracted by GIST [3] [4] and using fixed-size square images [8] conducted on the commonly employed machine learning models, i.e. *k*-NN, SVM and CNN.

4.1 Image Normalization and Sizing down

For classification, all malware should be normalized to be the same size [28] [7] [8]. This is because classifiers are unable to find the similarities or discrepancies among images if their sizes are different. Moreover, machine learning models (e.g. *k*-NN, SVM, CNN) have a fixed structure with a given number of inputs and outputs. Therefore, the size of the input images should be fixed to match the number of inputs of the employed model.

In other related works [3] [4] [6] [7] [8], in order to classify the malware, the input images are normalized to be a fixed size square image [8] such as 64×64 as illustrated in Fig. 7 (a).

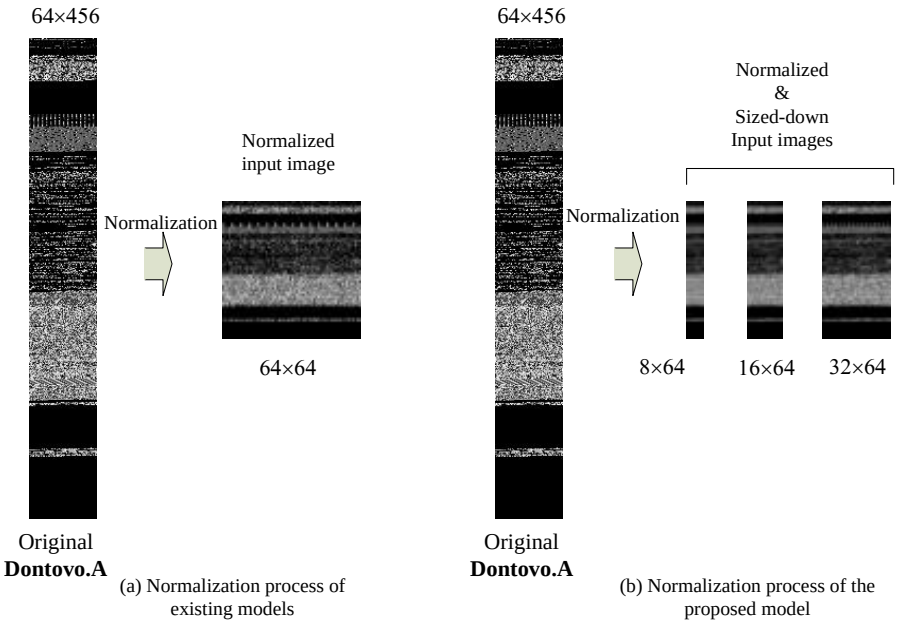


Fig. 7. Normalization process of (a) existing models and (b) the proposed model on Dontova.A malware

As discussed in Section II, the texture of a malware image is mainly distributed in vertical orientation. Hence, it is possible to decrease the dimension of the input malware images by sizing down their width,

e.g. 32×64, 16×64 or even 8×64 pixels instead of a square image as done in the literature (see Fig. 7 (b)). In doing so, the malware image is re-sampled with a given K factor mentioned in Eps. (9) and (10). In practice, K is selected such that the width of the original image is divisible by K, e.g. K = 2, 4, 8 for 32-pixel-width, 64-pixel-width and 128-pixel-width images. Having sized down the width, texture of the resized malware image is still retained (see also depicted Fig. 5). It helps to faster train the proposed model than others while still providing a high accuracy for classification.

It is acknowledged that in the proposed model grayscale pixel values are treated as the features of the malware image, which are applied to the training model. Therefore, the size of the input malware image greatly impacts the training time of the model. The bigger the input image size is, the longer the training time is.

4.2 The Employed Machine Learning Models

This paper applies k-NN as a machine learning technique for evaluating the proposed model. The k-NN is known as one of the simplest and most intuitive techniques among machine learning techniques [29]. It is based on the distance between the new sample (need to be classified) and k nearest neighbours (in the training set of observed data) for classification. The Euclidean distance is commonly used to determine the distance among samples for k-NN. Even though it is a simple model, several studies applied it for image-based malware classification [3] [4] and obtained positive results.

In this study, the training set is an n-dimensional vector X which contains a set of vectors of malware images I as mentioned in Section 3.1 and a vector y that contains their corresponding classes, e.g. Trojan, Backdoor, Worm. With a given k, an unknown variant x_0 gets assigned to the class c with the largest probability P determined as follows

$$P(\mathrm{y} = c \mid X = x_0) = \frac{1}{k} \sum_{i \in N} I(y_i = c) \tag{12}$$

where N is the number of variant samples in the training set, $I(y_i=c)$ is an indicator variable that evaluates to 1 if a given observation (x_i,y_i) in N is a member of class c, and 0 if otherwise.

The SVM is another simple algorithm in the field of machine learning [2] applied in this work. It is first designed for binary classification to which it searches for a hyperplane to separate two classes with the maximum margin, but the later versions of SVM can handle multi-class classification. One of the advantages of using the SVM is capable to solve high-dimensional dataset without overfitting. SVM is known to be more precise than k-NN [2], but it requires more time for training than k-NN. The linear hyperplane is commonly used for solving the classification problem by using SVM defined as follows [30]

$$wX + b = 0 \tag{13}$$

where X is training vector (containing a set of vectors of malware images I).

After identifying the hyperplane, the class of an unknown variant x_0 is classified as

$$class(x_0) = \mathrm{sign}(\mathrm{w}x_0 + b) \tag{14}$$

This work also applies CNN for image-based malware classification. CNN is one of the most commonly used deep learning architectures, especially for computer vision [31] and natural language processing [32]. As depicted in Fig. 8, a CNN applied for this paper has three different layers [33] which are convolutional, pooling and fully connected layers used for training and finding weights and biases that minimize certain loss function in order to map the inputs to the outputs as expected. The number of convolutional, pooling and fully connected layers can be more than one depending on the expected outcomes of designers.

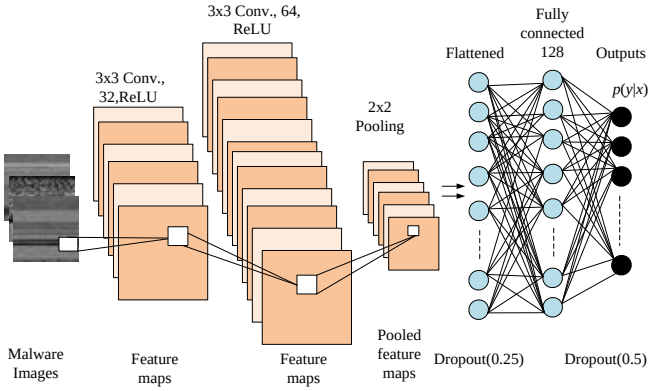


Fig. 8. The employed CNN architecture

The convolutional layer comprised of a set of learnable kernels (or filters) is to learn and build the *feature map* from input images. To map features into the feature map, several non-linear activation functions are employed, but the Rectified Linear Units (ReLU) is commonly used, i.e. ReLU[$f(x)$ = max (0, x)] where f is the objective function [33] [7]. Let ($H \times H$) be the size of the input image, ($F \times F$) be the size of the filter, and S be stride, the output of the feature map, W is given by

$$W = \left\lfloor \frac{H - F}{S} \right\rfloor + 1 \tag{15}$$

This paper applies a CNN architecture consisting of two convolution layers (as shown in Fig. 8) since the input images are very simple and in a small size. By doing so, the output of feature map is big enough for classification (see also Eq. (4)).

The next layer, the pooling layer (so-called down-sampling layer) shrinks down the resolution of the feature maps by splitting inputs into disjoint regions with size ($R \times R$) to generate it output for each region. Max-based or average-based techniques can be applied for pooling. If the input size is ($W \times W$) feeding for pooling, the output size is determined by

$$P = \left\lfloor \frac{W}{R} \right\rfloor \tag{16}$$

In our proposed model, pooling is constructed by applying the max-based technique with pool size (2×2). This helps to reduce half of the resolution and speed up the classification.

The last layer, a fully-connected layer including 128 neurons is to extract the global features from the inputs, which is usually a soft-max classifier, to estimate the posterior probability of each class label, y_i over m classes as following

$$y_i = \frac{\exp(-z_i)}{\sum_{j=i}^{m} \exp(z_j)} \tag{17}$$

In our experiment, to avoid *overfitting* caused by bad learning of the CNN, the dropout technique [34] is applied as. To do so, it shoots random neurons connected in each training iteration with a given probability, i.e. 0.25 and 0.5 before and after the fully connected layer (see also Fig. 8).

5 Experiments and Results

5.1 Datasets

This work adopts the Malimg [3] and Malheur [35] datasets and applies them for training and testing the proposed model. Those datasets are commonly used in the literature [3] [4] [5] [7] [8] [35]. The first dataset, the Malimg contains 25 malware families with 9,339 malware images (see TABLE 2). Those

malware images have *already* been converted from malware binaries. The second one, Malheur (see TABLE 3) contains 24 malware families with 3133 malware binaries but they have *not* been converted into images. In this study, they are converted into images by following the rule described in TABLE 1.

TABLE 2. MALIMG DATASET

No.	Family Name	No. Variants	Average Size (KB)
1	Allaple.L	1591	55
2	Allaple.A	2949	67
3	Yuner.A	800	328
4	Lolyda.AA 1	213	18
5	Lolyda.AA 2	184	17
6	Lolyda.AA 3	123	20
7	C2Lop.P	146	309
8	C2Lop.gen!G	200	443
9	Instantaccess	431	161
10	Swizzor.gen!I	132	237
11	Swizzor.gen!E	128	262
12	VB.AT	408	299
13	Fakerean	381	99
14	Alueron.gen!J	198	86
15	Malex.gen!J	136	70
16	Lolyda.AT	159	23
17	Adialer.C	125	129
18	Wintrim.BX	97	383
19	Dialplatform.B	177	12
20	Dontovo.A	162	16
21	Obfuscator.AD	142	148
22	Agent.FYI	116	11
23	Autorun.K	106	328
24	Rbot!gen	158	141
25	Skintrim.N	80	165
	Total	**9,339**	

TABLE 3. MALHEUR DATASET

No.	Family Name	No. Variants	Average Size (KB)
1	Adultbrowser	262	12
2	Allaple	300	24
3	Bancos	48	76
4	Casino	140	6
5	Dorfdo	65	4
6	Ejik	168	4
7	Flystudio	33	7
8	Ldpinch	43	4
9	Looper	209	17
10	Magiccasino	174	4
11	Podnuha	300	4
12	Posion	26	5
13	Porndialer	98	9
14	Rbot	101	4
15	Rotator	300	106
16	Sality	85	18
17	Spygames	139	4
18	Swizzor	78	27
19	Vapsup	45	12
20	Vikingdll	158	15
21	Vikingdz	68	31
22	Virut	202	4
23	Woikoiner	50	8
24	Zhelatin	41	19
	Total	**3,133**	

5.2 Experimental Environment & Setup

This experiment has been conducted on a Linux machine running Ubuntu 18.04 operating system with two processors of Intel Xeon 3.3GHz, 16GB RAM.

The machine learning models, i.e. *k*-NN, SVM and CNN is built on the platforms of Tensorflow [36] (an open-source software library developed by Google), Sci-kit learn [37] (supported by INRIA research institute) and Keras [38] (a popular library for deep learning in Python). Those platforms are widely used for both research and production [39] [40], thus they are reliable and adopted by research community.

This paper applies the commonly used resampling techniques, Bilinear for resizing down (down-sampling) the malware image.

5.3 Evaluation metrics

To evaluate the achieved results, some of following metrics are utilised [41] [42]

Accuracy: it is the total number of samples that are correctly predicted over the total number of samples. It is defined as follows

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of samples}} = \frac{TP + TN}{\text{Total numner of samples}} \quad (18)$$

where *TP* is true positive, *TN* is true negative (please refer to the paper [41] for further explanation of these parameters) .

Training time: it is the time taken for training the model. It is used for evaluation of training process of the proposed model to that of machine learning models feeding by fixed-size square images or GIST features.

5.4 Experiment Results

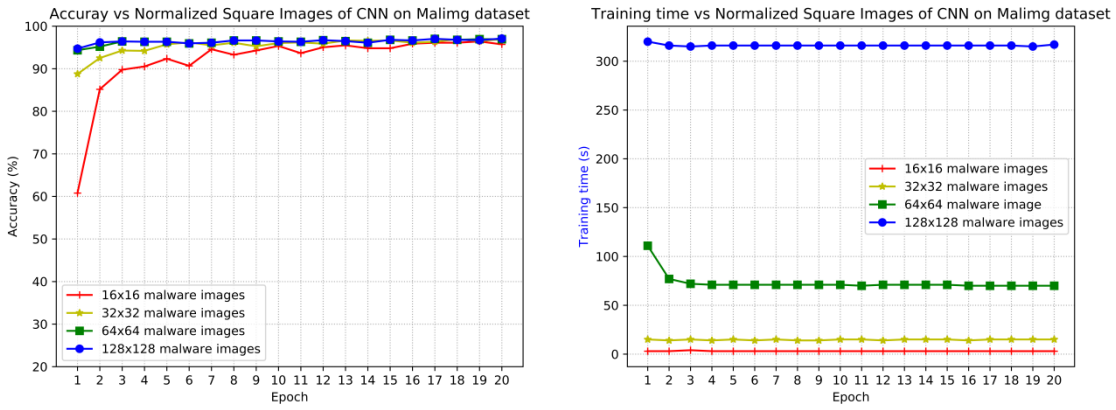
To highlight the enhancement of the proposed model, the experiment results of the proposed model (malware images are normalized into 32×64, 16×64, 8×64 pixels) are compared to the existing models in the literature in which input images are normalized by fixed-size square images of 64×64 pixels or using features provided by GIST.

5.4.1 The impact of normalized image size

It is noted that the size of the original input images are *not* the same. As shown in the TABLE 2 and TABLE 3, the average sizes of variants of different families (i.e. Allaple, Lolyda, Swizzor, etc.) are totally different, therefore they need to be normalized to the same size before training a machine learning model (see also Section 4.1).

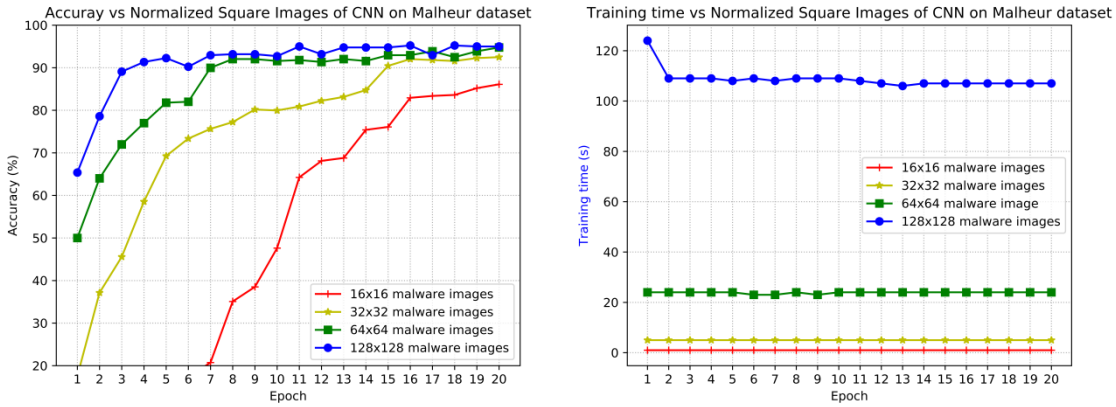
In this regards, one of the main concerns is to determine the size of the normalised image. It is chosen such that it is sufficient information (or features) for classification. Most previous research studies have normalized the input images into a fixed-size square image of 64×64 pixels which is equal to 64×64 bytes (or 4KB) [3] [4] [7] [8] [43] [15]. Our experiment show that the CNN trained by 64×64 images of Malimg and Malheur datasets converges to the highest accuracy (i.e. ~97% and 91% respectively) in 10 epochs as illustrated in Fig. 9(a) and Fig.10 (a). Applying 128×128 images for training enables the classifier to reach the same accuracy in 10 epochs but it takes a longer time for training than that of using 64×64 images (see Fig. 9(b)) and Fig.10(b)).

The experimental results in Fig.10 (a) also implies that the classifier needs a long time for learning to reach (or even never reach) the expected accuracy (~91%) if the training set contains many small-size-malware samples (i.e. 4 KB) as seen in TABLE 3. This is because the learning model (CNN) has not been provided sufficient information for learning when the input malware image is normalized to a smaller size (i.e. 32×32 or 16×16) than the original one (64×64 bytes or 4KB).



(a) (b)

Fig. 9. The (a) accuracy and (b) training time of the CNN with different square input images on Malimg dataset



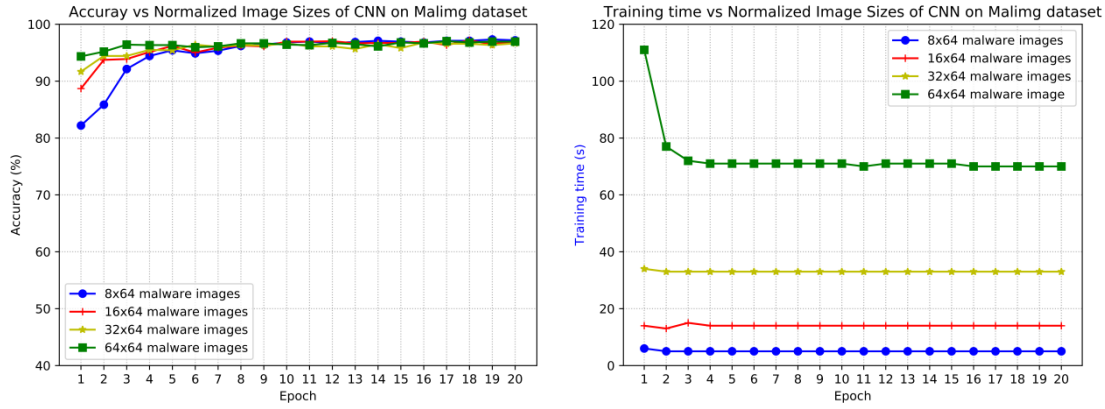
(a) (b)

Fig.10. The (a) accuracy and (b) training time of the CNN with different square input images on Malheur dataset

It is well-known that the size of recent malware, especially IoT malware, is reported to be reduced to a small one so that it can easily reside into a limited computational device such as a camera or a smart TV [17] [16]. Therefore, reducing the computational complexity and training time is important for those devices when training/re-training the classification model. One of solutions as proposed in this paper is to decrease the dimension of the normalized input images, i.e. shrink down the width of the input images but not too much impacting on the accuracy of classification. It enables the machine learning based classifier integrated in limited computational resources such as IoT devices to locally train/re-train the model instead of remotely training on the server as proposed in [15].

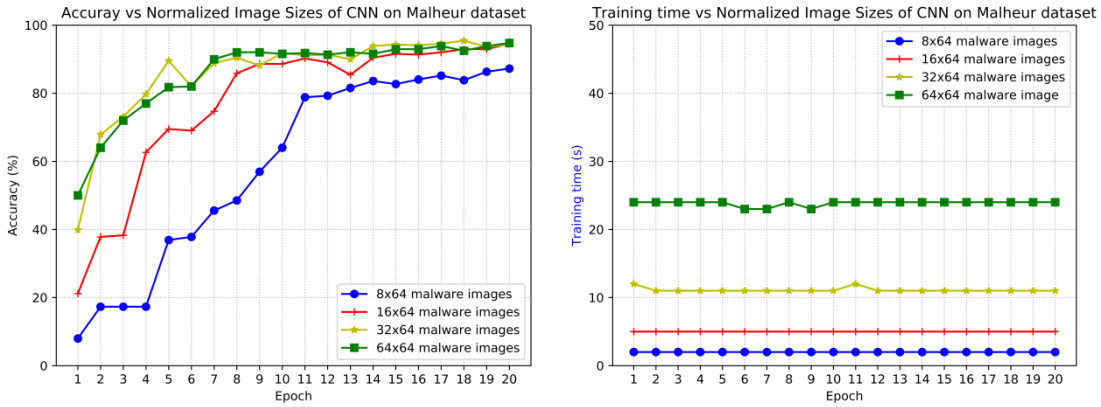
This experiment shows that the low dimension normalized input images (i.e. 32×64, 16×64, 8×64) still enable a classifier to produce an accurate classification as similar as the high dimension normalized input images (i.e. 64×64) do.

As depicted in Fig. 11(a) and Fig. 12(a) the accuracy of the proposed CNN model reaches 97.3% and ~91% after 10 epochs of training for four image sizes, i.e.64×64, 32×64, 16×64 on the Malimg and Malheur datasets respectively. Even when the width of the malware images reduces to one-eighth (i.e. 8×64) compared the original one after normalizing (i.e. 64×64 image), the accuracy of the CNN still remains almost the same as that of original one after 10 epochs on the Malimg dataset.



(a) (b)

Fig. 11. The (a) accuracy and (b) training time of the CNN with different sized-down input images on Malimg dataset



(a) (b)

Fig. 12. The (a) accuracy and (b) training time of the CNN with different sized-down input images on Malheur dataset

However, if the size of the input malware image is too small, the CNN classifier produces more losses than that of large one. As shown TABLE 3, many malware samples of the Malheur dataset have a small size, i.e. 4 KB, compared to the Malimg's malware samples, hence, the accuracy of classification obtained on this dataset is less than that of Malimg dataset, i.e. ~91% and below. Especially for 8×64 normalized images as illustrated in Fig. 12(b), the accuracy is less likely to reach the expected value of 91%. This is because the CNN has not been provided sufficient information for learning if the input images are normalized to a small size (i.e. 8×64 pixels). Therefore, this paper would recommend that the size of the normalized image should not be reduced less than 8×64 pixels if the original size of the input images (or input binaries) is too small (i.e. 4KB or below).

It is observed on other classifiers (i.e. *k*-NN and SVM) that the accuracy of classification almost reaches the expected value when training by lower dimension images of 64×64, 32×64, 16×64 or even 8×64 pixels as depicted in TABLE 4 and TABLE 5 on both Malimg and Malheur datasets. These experiments repeat 10 times with different seed numbers to make sure that the dataset is shuffled before splitting it into the training set and the testing set.

Once again, as depicted in TABLE 5 regarding Malheur dataset, if the input images are normalized to a very lower dimension, i.e. 8×64 pixels, than the original ones, the accuracy of classification is observed to be significantly reduced, i.e. 90.45 and 80.6% by using *k*-NN and SVM respectively.

It is also observed in TABLE 4 and TABLE 5 that the accuracy of the k-NN model trained on small image sizes (i.e. 16×64, 8×64) is even higher than that of big image sizes. This phenomenon is called "'the curse of dimensionality" [44] caused by an exponential growth of volume associated with adding more dimensions to Euclidean space. In particular, for the image processing using the *k*-NN, the bigger the size, the more data points to compare to decide the neighbour, thus, the *k*-NN is said to suffer low performances with high dimensional data. There are some solutions proposed for this problem in the literature including dimensionality reduction or feature selection [45], but they are out of the scope of this paper.

To obtained results shown in TABLE 4 and TABLE 5, it is noted that the malware images are fed directly into *k*-NN and SVM without the use of any image descriptor.

TABLE 4. A COMPARISON OF THE PROPOSED MODEL ON MALIMG DATASET USING *K*-NN AND SVM

	Normalized image size	Accuracy		Training time (s)
		Average (%)	Std(*)	
k-NN	64×64	97.38	3.747	2.69
k-NN	32×64	97.98	0.27	1.22
k-NN	16×64	98.08	0.28	0.58
k-NN	8×64	98.11	0.25	0.25
SVM	64×64	98.49	0.16	35.72

SVM	32×64	98.51	0.18	11.93
SVM	16×64	97.39	0.33	4.16
SVM	8×64	97.41	0.32	1.61

(*) Std: standard deviation

TABLE 5. A COMPARISON OF THE PROPOSED MODEL ON MALHEUR DATASET USING *K*-NN AND SVM

	Normalized image size	Accuracy		Training time (s)
		Average (%)	Std	
k-NN	64×64	91.28	1.52	0.65
k-NN	32×64	91.32	0.66	0.36
k-NN	16×64	94.1	0.62	0.13
k-NN	8×64	90.45	0.61	0.06
SVM	64×64	95.79	0.62	10.84
SVM	32×64	94.13	0.74	4.27
SVM	16×64	91.40	0.77	2.03
SVM	8×64	80.06	1.28	1.22

Through experiments conducted on two datasets (Malimg and Malheur), it turns out that the width of the malware images has less significance than the height since the texture vertically spread across the image as discussed in Section III-B. Hence, it can be able to re-size down the width of the malware image to reduce the complexity and training time (as determined in Eqs. (12), (13) and (15)) while still providing a high accuracy for classification. This finding is significant for training of those devices which have limited computational resources such as IoT devices.

5.4.2 A comparison of the proposed model with GIST-based feature model

It is well-known that the GIST descriptor provides a low dimension of the scene by summarizing the gradient information (scales and orientations) for different parts of an image [11]. Therefore, some related studies in the literature have applied GIST features for training their image-based malware classification models such as [3] [5]. For comparison, this paper applies the same GIST descriptor as used in [3] -and [5] for which it extracts 320 features for each variant sample. It is noted that in the proposed model, pixels of malware images are treated as features. Therefore the dimension of input images used for training of the CNN should be normalized to the same number of features provided by GIST, i.e. 320 pixels (or 10×32 image). The comparison is conducted on Malimg dataset in two schemes:

i. Non-deep learning models (*k*-NN, SVM) fed by GIST features vs. fed by malware images. The size of normalized malware images is of10×32 pixels and a smaller one, i.e. 8×32 pixels.

ii. Deep learning model (i.e. CNN) trained by malware images of 10×32 pixels. This is because a CNN can build its own features from the input raw images, feeding extracted features to a CNN is not necessary.

TABLE 6. A COMPARISON OF THE PROPOSED MODEL VS. GIST-BASED FEATURE MODEL

Machine learning model	Normalized image size	Accuracy		Training time (s)	Remark
		Average (%)	Std		
k-NN + GIST features	n/a	97.06	0.26	0.138	Fed by 320 GIST features
k-NN + proposed model	10×32	98.31	0.28	0.146	
k-NN + proposed model	8 × 32	98.34	0.25	0.11	
SVM + GIST features	n/a	95.54	0.396	2.59	Fed by 320 GIST features
SVM + proposed model	10×32	97.27	0.26	1.11	
SVM + proposed model	8×32	97.08	0.28	0.93	
CNN (10 epochs)	10×32	96.08	0.45	3.0	

To obtain the results illustrated in TABLE 6, each experiment was performed 10 times with different seed numbers to ensure that the dataset is shuffled prior to splitting it into the training set and the testing set. As shown in TABLE 6, the *k*-NN and SVM trained by re-sized down images outperform GIST-based *k*-NN and SVM models. Even the size of malware images are normalized at 8×32 pixels which is lower than the 320 pixels, the accuracy achieved by the proposed model using *k*-NN and SVM is observed higher than that of GIST-based models, i.e. 98.34% and 97.8% (instead of 97.6% and 95.54% respectively). The CNN achieves 96.08 % of accuracy which is almost the same compare to that of GIST-based *k*-NN and SVM models, i.e. 97.06% and 95.54% respectively but it does not need GIST features for training.

The comparison presented in TABLE 6, once again, turns out that it is capable of resizing down the width of the input malware images before training a machine learning-based classifier to reduce the computational complexity and training time.

6 CONCLUSIONS

This paper proposed a simple and effective image-based malware classification model using machine learning. The novel idea is to feed raw images whose size is horizontally reduced (i.e. the width of the image) to train machine learning models, i.e. *k*-NN, SVM and CNN, instead of training by fixed-size square image or GIST features as existing models do. This is because the texture of the malware image is found that mainly spread in the vertical orientation as analyzed in this paper. Hence, the horizontal size of the malware image is able to be reduced without too much impact on the texture. This finding helps to significantly decrease the computational complexity and training time of the machine learning models applied for image-based malware classification. The proposed model, therefore, is useful for training the classifier in those devices which have limited computational resources such as IoT devices.

Our experiment conducted on two datasets, i.e. Malimg and Malheur, shows that when the size of the input images reduces from the fixed-sized square one, i.e. 64×64 to 32×64, 16×64, or even 8×64, the accuracy of the proposed model keeps almost the same (i.e. 97% on Malimg dataset and ~91% on Malheur dataset) after 10 epochs of training. Whereas the training time of the proposed model reduces a half, a quarter, and one-eighth respectively compared to training time taken by the machine learning-based classifier (i.e. *k*-NN, SVM and CNN) feeding by fixed-sized square image, i.e. 64×64. Also, the accuracy of the proposed model is higher than that of the GIST-based malware classification model without taking time for extracting GIST features.

ACKNOWLEDGMENT

Chando Lee would like to thank National IT Industry Promotion Agency (NIPA), Korea, for its funding. The research was done during his term as a World Friends Korea Advisor at Vietnam – Korea University of Information and Communication Technology, Da Nang City, Vietnam.

REFERENCES

[1] 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. 3, pp. 1-22, 2018.

[2] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," *ACM Computing Surveys (CSUR),* vol. 50, no. 3, p. Article No. 41, 2017.

[3] L. Nataraj, S. Karthikeyan, G. Jacob and B. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, Pittsburgh, Pennsylvania, USA, July 20 - 20, 2011.

[4] N. Bhodia, P. Prajapati, F. D. Troia and M. Stamp, "Transfer Learning for Image-based Malware Classification," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy, pp.719-726*, 2015.

[5] S. Yajamanam, V. Selvin, F. D. Troia and M. Stamp, "Deep Learning versus Gist Descriptors for Image-based Malware Classification," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pp. 553-561*, 2018.

[6] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digital Investigation,* vol. 26, no. 1, pp. 5118-5126, 2018.

[7] S. Venkatraman, M. Alazab,and R.Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications,* vol. Volume 47, pp. pp. 377-389, August 2019.

[8] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 7, pp. 3187-3196, 2018.

[9] The European Union Agency for Cybersecurity (ENISA), "ENISA Threat Landscape 2021," 27 Oct 2021. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021.

[10] Kaspersky, "Kaspersky Security Bulletin 2021. Statistics," 15 Dec 2012. [Online]. Available: https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf.

[11] "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *International Journal of Computer Vision,* vol. 42, no. 3, pp. 145-175, 2001.

[12] S. A. K. Tareen and Z. Saleem, "A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *International Conference on Computing, Mathematics and Engineering Technologies (iCoMET 2018)*, Sukkur, Pakistan, 2018.

[13] "Kaspersky Security Bulletin 2019," Kaspersky, 2019. [Online]. Available: https://securelist.com/kaspersky-security-bulletin-threat-predictions-for-2019/88878/.

[14] Kaspersky, "Kaspersky Security Bulletin 2020 Statistics," 2020. [Online]. Available: https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2020_en.pdf.

[15] J. Su et al., "Lightweight Classification of IoT Malware Based on Image Recognition," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, 2018.

[16] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer* , vol. 50, no. 7, pp. 80-84, 2017.

[17] "A survey of IoT malware and detection methods based on static features," *ICT Express,* vol. 6, no. 4, pp. 280-286, 2020.

[18] 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.

[19] Z. Lv, "Security of Internet of Things edge devices," *Special Issue:Resource management of IoT edge devices: Challenges, techniques, and solutions,* vol. 51, no. 12, pp. 2446-2456, 2021.

[20] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, Cordelia Schmid, "Evaluation of GIST descriptors for web-scale image search," in *Proceedings of the ACM International Conference on Image and Video Retrieval, Article No. 19*, Santorini, Fira, Greece, 2009.

[21] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov and G. Giacinto, "Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, Louisiana, USA, 2016.

[22] T. Alex, "Malware-detection-using-Machine-Learning," [Online]. Available: https://github.com/tuff96/Malware-detection-using-Machine-Learning.

[23] M. Jain, W. Andreopoulos and M. Stamp, "Convolutional neural networks and extreme learningmachines for malware classification," *Journal of Computer Virology and Hacking Techniques,* vol. 16, pp. 229-244, 2020.

[24] R. M. Haralick, K. Shanmugam, I-H. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, Vols. SMC-3, no. 6, pp. 610 - 621, 1973.

[25] W. R. Tobler, "A Computer Movie Simulating Urban Growth in the Detroit Region," *Economic Geography,* vol. 46, no. Supplement: Proceedings. International Geographical Union. Commission on Quantitative Methods, pp. 234-240, 1970.

[26] B. Jähne, Digital Image Processing, Springer Berlin Heidelberg, ISBN 3-540-24035-7 , 2005.

[27] A. C. Bovik, The Essential Guide to Image Processing, Academic Press, 1st edition, ISBN-13: 978-0123744579, 2009.

[28] D. J. Hemanth and V. V. Estrela, Deep Learning for Image Processing, IOS Press, ISBN 978-1-61499-821-1, 2017.

[29] J. Orava, "k-nearst Neighbour Kernel Density Esimation, the Choice of Optimal k," *Tatra Mountains Mathematical Publications,* vol. 50, no. 1, pp. 39-50, 2011.

[30] A. W. Moore, "Tutorial slides by Andrew Moore," [Online]. Available: http://www.cs.cmu.edu/~awm. [Accessed 24 June 2021].

[31] C. Farabet, C. Couprie, L. Najman and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, pp. 1915-1929, 2013.

[32] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Maryland, USA, pp. 655–665, 2014.

[33] S. Albelwi and A. Mahmood, "A Framework for Designing the Architectures of Deep Convolutional Neural Networks," *Entropy,* vol. 19, no. 6, Article 242, 2017.

[34] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Journal of Neural Networks,* vol. 71, no. C, pp. 1-20, 2015.

[35] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz., "Automatic Analysis of Malware Behavior using Machine Learning," *Journal of Computer Security (JCS),* vol. 19, no. 4, pp. 639-668, 2011.

[36] Google Brain Team, "TensorFlow," [Online]. Available: https://www.tensorflow.org/. [Accessed 18 Nov 2019].

[37] F. Pedregosa et al., "Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825--2830, 2011.

[38] Keras, "Keras Documentation," 27 March 2015. [Online]. Available: https://keras.io/.

[39] M. Abadi et al., "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, Savannah, GA, USA, pp. 265-283, 2016.

[40] J. Van den Bossche et al., “Scikit-learn,” [Online]. Available: https://scikit-learn.org/stable/. [Accessed 18 Nov 2019].

[41] D. M. W. Powers, “Evaluation: From Precision, Recall And F-Measure To Roc, Informedness, Markedness & Correlation,” *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011.

[42] M. Stamp, “Data Analysis,” in *Introduction to Machine Learning with Applications in Information Security*, ISBN-13: 978-1-138-62678-2, CRC Press, Taylor & Francis Group, 2018.

[43] K. S. Han, J. H. Lim , B. Kang and E. G. Im, “Malware analysis using visualized images and entropy graphs,” *International Journal of Information Security*, vol. 14, no. 1, pp. 1-14, 2015.

[44] R. E. Bellman, Adaptive Control Processes: A Guided Tour, Princeton: Princeton University Press, 1961.

[45] E. Keogh, A. Mueen, “Curse of Dimensionality,” in *Encyclopedia of Machine Learning* , Boston, MA, Springer, ISBN 978-0-387-30164-8, 2011.