An Attention Mechanism for Combination of CNN and VAE for Image-Based Malware Classification

TUAN VAN DAO, HIROSHI SATO, AND MASAO KUBO
Department of Computer Science, National Defense Academy, Yokosuka, Kanagawa 239-8686, Japan
Corresponding author: Tuan Van Dao (ed21006@nda.ac.jp)

ABSTRACT
Currently, malware is increasing in both number and complexity dramatically. Several techniques and methodologies have been proposed to detect and neutralize malicious software. However, traditional methods based on the signatures or behaviors of malware often require considerable computational time and resources for feature engineering. Recent studies have applied machine learning to the problems of identifying and classifying malware families. Combining many state-of-the-art techniques has become popular but choosing the appropriate combination with high efficiency is still a problem. The classification performance has been significantly improved using complex neural network architectures. However, the more complex the network, the more resources it requires. This paper proposes a novel lightweight architecture by combining small Convolutional Neural Networks and advanced Variational Autoencoder, enhanced by channel and spatial attention mechanisms. We achieve overperformance and sufficient time through various experiments compared to other cutting-edge techniques using unbalanced and balanced Malimg datasets.

INDEX TERMS Malware classification, variational autoencoder, channel attention, spatial attention, latent representation, information security.

I. INTRODUCTION
The Internet has become an essential function in our lives. However, at the same time, it also raises many security threats while providing excellent service. Malware is a powerful tool for an attacker to intrude, sabotage, and control targets indirectly as a remote administration tool through the Internet. The abuse of various malware causes a significant impact on cyber-security and threats to individuals, society, and countries [1], [2]. Authors of malware mix different evading techniques such as user interaction, environment awareness, obfuscation, code compression, and code encryption to change existing malicious code’s appearance to bypass the Anti-virus System and Intrusion Detection System (IDS). However, it is often the case that the new variants still have the same malicious intentions and characteristics as the original malware.

There are two malware detection and analysis techniques: static analysis and dynamic analysis. The static analysis investigates the malware without executing them [3]. This type of analysis utilizes various information, such as Application Programming Interface (API) calls, the entropy of files, and strings that are all embedded in raw bytes of the Portable Executable (PE) [4]. The main limitation of static analysis is that it is not sufficient in the case of code obfuscation and zero-malware. In addition, the analysis will be time-consuming if malware is mixed up with many disruptive methods.

On the other hand, dynamic analysis investigates the malware as they are executed in simulated environments like sandboxes or virtual machines [5]. This analysis does not require disassembling the PE file and decompressing as well as unpacking in advance to gain the malware’s features as static analysis. The main limitation of this analysis is that the dynamic analysis may not always uncover malicious behavior because some malware can detect virtual environments and change its behavior. Moreover, because of the rapid development of many automatic malware creation tools [6], these methods cannot catch up to the speed of malware generation.
Machine learning has become more potent because its highly developed algorithms can solve most problems encountered in almost every field. Several methods extract elements from malicious software, such as API calls [7], [8], and feed them into machine learning. Some of them take advantage of Natural Language Processing (NLP) to solve strings element for detection [9] and classification tasks [10]. Existing malware classification researches use machine learning techniques like Support Vector Machine (SVM) [11], K-Nearest Neighbor [12], and Random Forest [13].

Another alternative to the machine learning-based method for malware classification is the vision-based approach [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39]. Although attackers use obfuscation techniques to achieve spoofing, malware variants from the same family still maintain similar code and data order, which may not appear in the same location. The convolutional Neural Networks (CNN) can extract common features from a family. Conti et al. have proposed a method to visualize malware binaries into a grayscale image and noticed that visual analyses of malware binary help distinguish various regions of data from the image [15]. The advantage of the malware visualization analysis is that it does not require using any decompilers or a dynamic running environment. Moreover, malware samples are converted into RGB (Red, Green, Blue) in [16] by encoding and arranging bytes from binary files. A color image can obtain more information than a grayscale image.

The growth of high-performance computing, coupled with the enormous CNNs architectures, made it possible to process images at a higher level of complexity. However, recent studies indicate that fewer parameters with a simple network structure give relatively satisfactory results and can be applied to low-profile devices like IoT [17] or smartphones [18].

Taking advances from well-known CNN architectures, Transfer learning is also applied for image-based malware classification [19], [24], [25], [28], [30]. By using pre-trained CNNs and fine-tuning them, several CNNs can extract rich features more than simple ones [19].

Another approach that can be used to extract features of an image is Autoencoder (AE). AE is an unsupervised deep learning algorithm with a unique neural network structure. AE transforms the input into an output with minimal reconstruction errors and can process with small data. However, AE often falls into overfitting, and the problem of organizing the latent space is complex. VAE is then introduced as an autoencoder whose training is regularized to avoid overfitting and ensure that the latent space has suitable properties that enable a generative process. While VAE can represent global features through latent space, CNN capture local feature through small kernels. The combination of VAE and CNN promises to obtain an overall feature of the object [32]. However, this combination still did not achieve the expected performance.

For now, attention mechanisms [20] have been a significant breakthrough in deep learning. The mechanisms have been widely used in image recognition, NLP, and speech recognition. However, few studies on malware classification are based on attention mechanisms in computer vision. Moreover, compared to multi-head attention [20], this type of attention tends to feedforward CNN and can be applied at every convolutional block in deep networks.

So far, recent studies have focused mainly on the depth and width of Neural Networks and increase amount of features but have not yet focused on enriching the quantity of object features. This paper aims to gather as many worthwhile features as possible while keeping small model architecture by utilizing CNN and combining it with a new type of Variational Autoencoder enhanced by the Attention mechanism, which we call “AVAE”. The AVAE can provide more discriminate features, map and refine the original feature space to latent representation.

The main contribution of this paper is providing an image-based malware classification system through feature synthesis from VAE, CNN, and attention mechanism. Because the processing is merely dependent on images, the system does not require in-depth knowledge of the malware and the environment to determine its behavior. Moreover, some classifiers can give the result in under a second, so our model can be applied in real-time countermeasures against malware.

The rest of the paper is organized as follows: Section II discusses the related work concerning some popular and recent techniques in malware detection and classification. Section III illustrates the proposed model in detail. Section IV evaluates the performance of the proposed approach. Finally, we summarize our work in Section V.

II. RELATED WORK

In this session, we investigate various new studies on image-based malware classification, ranging from models with simple structures to complex ones; some hybrid models with different structural combinations have achieved high performance in malware classification.

For the first time, Nataraj et al. proposed a novel approach for visualizing and classifying malware using image processing techniques [12]. They visualized malware as a grayscale image based on the observation that images of the same class were very similar in layout and texture. They utilize GIST descriptor, based on wavelet decomposition of an image, as feature extractor and k-nearest neighbor (kNN) as a classifier. The paper achieved an accuracy of 97.18% on their introduced dataset: Malimg, which contains 9,339 malware samples related to 25 different malware families. Other feature descriptors are also applied, such as HOG and HOC + GIST [22]. However, this method is not suitable for processing a massive amount of malware because of the high computational cost. Naeeem et al. [23] utilized a new type of feature descriptor by combining and balancing collective local and global feature vectors. As a result, they achieved a high classification rate of 98% on the Malimg dataset.
The current research focuses on building a complex network model with deep CNN. For example, more than ten Conv layers [2], VGG16 in [24], VGG19 in [25], or Combining multiple CNN architectures [19]. On the other hand, [26] minimize parameters to speed up training. The proposed model achieves the accuracy, which is lower, approximately under 1%, than the state-of-the-art performance by reducing 99.7% the number of trainable parameters of the best model in the comparison session.

Verma et al. [27] try to enrich extracted Malware features by concatenating CNN features and other 35 statistical texture features. The numerous CNNs require high-resolution images for training. The input image size of these networks is usually around 224 × 224 to 299 × 299 [28]. The larger the size, the higher the computational cost. Roseline et al. [29] employed Lightweight CNNs with merely three convolutional layers with an increasing depth of 16, 32 and 64. The model is optimized by Adam and utilizes Categorical Cross-entropy loss, and the input image is resized to 32 × 32. With the above setting, [29] achieved an accuracy of 97.68% through 50 epochs.

Rezende et al. [30] transferred the first 49 layers of ResNet-50 on ImageNet to the malware classification task. Frozen layers can be seen as learned feature extraction layers. The author replaced the last layer with 1000 fully connected softmax with 25 fully connected ones according to the number of classes on the Malimg dataset. After 750 epochs, the paper reached an average accuracy of 98.62% with 10-fold cross-validation. They also compare features extracted from Deep CNN (DCNN) with GIST features using the same kNN classifier. The experimental result showed ResNet-50 performed better than handcrafted GIST by 0.52% with 98.00% and 97.48%, respectively.

Vasan et al. [19] utilized an ensemble of CNNs. They assumed that different CNNs provide different semantic representations of the image; therefore, higher qualities feature is extracted than traditional methods. VGG16 and ResNet-50 pre-trained on ImageNet were fine-tuned for malware images. This ensemble method achieves high detection accuracy with a low false rate.

Anandhi et al. [21] introduced another type of Deep CNN with Densely connected networks (DensNet). DensNet comprises dense blocks, a composite function, and a transition layer. This architecture solved the vanishing gradient problem because of the shrinking of the gradient through a deep network. The author utilized DenseNet201 with 201 layers deep and achieved an accuracy of 98.97% on the original Malimg dataset and 99.36% by combining similar families, C2LOP and Swizzor.

Çayır et al. [26] proposed a simple architecture called Random CapsNet forest engineering instead of complex CNN architectures. This model contains capsules similar to autoencoders, with each capsule learning how to represent an instance for a given class. Although the proposed method does not use data augmentation, data resampling, transfer learning, and weighted loss function, it still achieved acceptable results with an accuracy of 98.72%.

Nisa et al. [31] combine the features extracted from pre-trained AlexNet and Inception-V3. These fusion features are then classified using different classifiers such as SVM, kNN, and Decision tree (DT). Reference [31] achieved an accuracy of 98.7% on the Malimg dataset. The result was improved up to 99.3% when applying augmentation to turn Malimg into a balanced dataset.

Lee and Lee [1] illustrates the effectiveness of autoencoder by applying multiple AEs. Each AE model classifies only one type of malware and is trained using only samples from the corresponding family. As a result, the author achieves an accuracy of 94.03% for a system with the same AE network structure and 97.75% with various AEs. Moreover, the model achieves a 0.46% improvement from 97.75% to 98.21% when combining similar classes. However, the article is still misclassified quite a lot, showing that AE has not been effective in extracting the characteristics of image-based malware.

Burks et al. [32] inserted VAE into the handcraft Residual Network (RN), and the performance accuracy of 85% increased by 2% and 6% compared with the original RN and Generative Adversarial Network (GAN) model, respectively.

Awan et al. [25] applied spatial convolutional attention called dynamic spatial convolution on VGG19 Network. This attention utilized a global average pooling (GAP) mechanism, rescale the output of GAP by lambda layer, fed into dropout of rate 0.25 before Fully connected layer, the author utilized Softmax as a traditional classifier of CNNs. The performance was evaluated on the Malimg dataset and achieved an accuracy of 97.68%. Ma et al. [33] applied the attention mechanism [20] and handcrafted architecture with five parts: Input layer, Local Attention, Global Attention layer, Dense layer, and Output layer. Compared with other methods, the combination of the attention mechanism and CNN mechanism achieved the best classification accuracy of 96.09% on Microsoft’s Kaggle dataset.

Narayanan et al. [42] declare that each malicious program belonging to a family has a distinct pattern. The authors use Principal Component Analysis (PCA) as linear dimension reduction can save the computational time and even trade-off of losing several valuable information. As a result, the performance obtained is still far behind CNN. Davuluru et al. [43] indicate a trade-off between computational time and model complexity. The authors also highlight the advantages of using CNN as a feature extractor. Instead of the original CNN classifier (softmax), using SVM can overcome the drawback of the limited unbalanced dataset.

Narayanan and Davuluru et al. [44] proposed a novel approach of fusing both Natural Language Processing (NLP)-based approach called LSTM and image-based approaches including simple CNN, AlexNet, ResNet, and VGG16 into
a single simple architecture. The combination of several different CNN feature extractors is also somewhat similar to the characteristics of the DensNet model, concatenating intermediate layers [21]. The authors extract 9 features from each of those architectures, compiling a suite of 45 in total. Choosing the appropriate features from the total number of features in each architecture will also become an optimization problem for two different architectures. Besides, recent malware is obfuscated, and the obtained opcodes sequence will be entangled with a lot of noise, leading to limitations in finding relationships between words and the quality of embedding of LSTM. As a result, it will affect the assembly architecture. Besides, observing malware visualization from the Microsoft Malware Classification Challenge (BIG 2015) dataset, it can seem that different families have distinctly different images that the naked eye can distinguish. The number of families is not too large; compared to the Malimg dataset, up to 25 families, several malware samples from different families look the same and cannot be distinguished by the human eye. Therefore, with data of higher complexity, an additional refine mechanism is needed; in this study, we focus on filtering and selecting essential features so that they can be processed with data with high similarity even if the naked eye cannot distinguish it.

### III. PROPOSED METHOD

#### A. IMAGE REPRESENTATION FOR MALWARE

To visualize a malware sample as an image, we must interpret every byte as one pixel in an image. Notice that binary files are the hexadecimal representation of the PE of malware in Figure 1. The first row is the offset of the memory address. The second one represents the pair of hexadecimal. Each hexadecimal pair is treated as a single decimal number which serves as a pixel value of the image. The resulting array must be organized as a 2-D array, and values are in the range [0, 255] (0: black, 255: white). The size of the image depends on the binary file’s size. Table 1 presents different heights for malware images due to different sizes of malware files while fixing the width of images. Table 1 also illustrates that converting malware into grayscale images does not require a long time; common malicious codes less than 1Mb in size only take no more than 0.01s to convert.

We then convert the grayscale images into three-channel RGB images by replicating the grayscale channels for three iterations. Figure 2 illustrates a part of the malware plot from the Malimg dataset, which Nataraj et al. [12] created. It can be observed that images from a given family are similar while distinct from those of a different family. New variants are often created by changing a small part of the code. Therefore, if the predecessor is reused, the result would be very similar. Furthermore, by converting malware into an image, it is possible to detect the small changes while keeping the comprehensive structure of samples belonging to the same family.

#### B. VARIATIONAL AUTOENCODER

VAE [34] is a variant of an autoencoder (AE) that also consists of an encoder and a decode. The autoencoder is solely trained to encode and decode with as few losses (reconstruction loss) as possible, no matter how the latent space is organized. Therefore, it is tough to guarantee that the encoder will organize the latent space smartly. More than that, AE often faces an overfitting problem which causes irregular in the latent space. On the other hand, the VAE applies a Gaussian probability density \( q_\phi(z|x) \) that makes the encoder return distribution over the latent space. VAE tackles the problem of the latent space irregularity problem by adding in the loss function a regularisation term over that returned distribution to ensure a better organization of the latent space.
The model can learn what and where to emphasize or suppress and refines intermediate features effectively through the attention mechanism, [40]. In this paper, we apply both CAM and SAM. It is called Convolutional Block Attention Module (CBAM) [40] in the encoder part of VAE. We name it as Attention of Variational Autoencoder (AVAE).

D. FEATURE COMBINATION AND CLASSIFICATION

Fig. 4 illustrates the architecture of our system. We utilize the lightweight CNN with merely two convolutional layers with a kernel size is 32, followed by 64. Before flattening the pooled feature map, we apply dropout with a rate $= 0.5$ to avoid overfitting. Moreover, we use Adam as a fine-tuning optimizer with a minimal learning-rate $= 0.001$. In the A VAE model, we insert CBAM in turn between convolutional layers. In latent representation, we use the mean vector, dense $\mu$ with latent dimension sets to 100. We concatenate these extracted features with a fully connected layer of CNN. Both the CNN model and A VAE model train low-resolution image with the size of $64 \times 64$, and the number of epochs is 50.

We utilize early stopping to finish training without improvement after five epochs. We use the typical classifiers algorithm of machine learning to evaluate our system. In order to evaluate our method, we utilize 10-fold Cross-Validation. One of the ten subsamples is held out as validation data, and the remaining nine subsamples are used as training data. This process is repeated ten times with each of the ten subsamples used as validation. The average of ten results is the quality of the method.

IV. EXPERIMENTAL RESULTS

A. DATASET

This study evaluates our model using the Malimg Dataset consisting of 9,339 malware samples of 25 different families. Table 2 illustrates the number of malwares in each class. It is clear that the Malimg dataset is unbalanced; 2,949 images represent the Allaple. A malware family, while merely 80 images are present in the Skintrim. N family. The imbalanced datasets are a communal problem in machine learning in general, and computer vision in particular [28], [35], [36].

Furthermore, imbalanced data harms the performance of the CNNs because of causing underfitting and overfitting [37]. There are two standard methods to deal with imbalanced class distribution problems; oversampling and undersampling. Instead of adding more samples on lacking malware families, [32] utilized image augmentation, which generates new data from classes with less population in the dataset. However, using augmentation is an extremely high computational cost. In this study, we adopt undersampling to balance the Malimg dataset. Specifically, we reduce the number of malware samples in all groups to the lowest sample Skintrim.N family same with [38]. The total number of variants now is less than one-fourth of 2,000 compared to the original Malimg dataset.

| Symbol | Meaning |
|--------|---------|
| $W$  | Weight matrix of encoder |
| $W'$ | Weight matrix of decoder |
| $b$  | Bias vector of encoder |
| $b'$ | Bias vector of decoder |
| $\phi$ | Parameter for training encoder |
| $\theta$ | Parameter for training decoder |
| $x$ | Training dataset |
| $z$ | Representation of the input sample |
| $x^i$ | datapoint $i$ th |
| $\varphi$ | Encoder |
| $q_0$ | Decoder |
| $l_{VAE}(x^i, \theta, \phi)$ | Loss function of VAE for a datapoint $x^i$ |
| $q$ | Deterministic function |
| $K$ | Number of samples that are utilized to reparameterize $z$ |

Let $\phi = (W, b)$ and $\theta = (W', b')$. The lost function of VAE includes two terms as follows: 

$$
\begin{align*}
    l_{VAE}(x^i, \theta, \phi) &= -E_{q_0(z|x^i)} \left[ \log p_\theta(x^i | z) \right] + D_{KL}(q_\phi(z) \parallel p(z)) \\
    &= -\frac{1}{K} \sum_{k=1}^{K} \log p_\theta(x^i | z^{i,k}) + D_{KL}(q_\phi(z) \parallel p(z))
\end{align*}
$$

(1)

The first term is the expected negative log-likelihood of the $i$-th data point. This term is also called the reconstruction error (RE) of VAE since it forces the decoder to learn to reconstruct the input data. The second term is the Kullback-Leibler (KL) divergence between the encoder’s distribution $q_\phi(z|x)$ and the expected distribution $p(z)$. This divergence measures the relation of $q$ and $p$ [34]. In the VAE, $p(z)$ is specified as a standard normal distribution with mean zero and standard deviation, denoted as $\mathcal{N}(0, 1)$. If the encoder outputs representations $z$ different from the standard normal distribution, it will receive a penalty in the loss. Since the gradient descent algorithm is not suitable to train a VAE with a random variable $z$ sampled from $p(z)$, the loss function of the VAE is re-parametrized as follows:

$$
\begin{align*}
    l_{VAE}(x^i, \theta, \phi) &= -\frac{1}{K} \sum_{k=1}^{K} \log p_\theta(x^i | z^{i,k}) + D_{KL}(q_\phi(z) \parallel p(z))
\end{align*}
$$

(2)

where $z^{i,k} = g_\phi(\epsilon^{i,k}, x^i)$, $\epsilon^{i,k}$ denotes $\mathcal{N}(0, 1)$.

After training, the latent layers of VAE can be utilized for a classification task. Then, the original data is passed through the encoder part of VAE to generate the latent representation.
TABLE 3. Performance comparison for the various classifier on unbalanced Malimg dataset.

| Class | Family name | No. of samples | Percentage(%) |
|-------|-------------|----------------|---------------|
| 0     | Adiater.C   | 122            | 1.31          |
| 1     | Agent.FY1   | 116            | 2.12          |
| 2     | Allapple.A  | 2849           | 31.58         |
| 3     | Appple.L    | 1591           | 17.04         |
| 4     | Aluereon.genJ| 198            | 2.12          |
| 5     | Autorun. K  | 106            | 1.14          |
| 6     | C2LOP.gen/g | 200            | 2.14          |
| 7     | C2LOP.P     | 146            | 1.56          |
| 8     | Dialplatform.B| 177          | 1.89          |
| 9     | Donuto.A    | 162            | 1.73          |
| 10    | Fakerean    | 381            | 4.08          |
| 11    | Instantaccess| 431            | 4.62          |
| 12    | Lolyda.AA1  | 213            | 2.28          |
| 13    | Lolyda.AA2  | 184            | 1.97          |
| 14    | Lolyda.AA3  | 123            | 1.32          |
| 15    | Lolyda.AT   | 159            | 1.70          |
| 16    | Malex.genJ  | 136            | 1.46          |
| 17    | Obfuscator.AD| 142            | 1.52          |
| 18    | Rbot.gen    | 158            | 1.69          |
| 19    | Skintrm.N   | 80             | 0.86          |
| 20    | Swizzor.gen/E| 128            | 1.37          |
| 21    | Swizzor.gen/I| 132            | 1.41          |
| 22    | VB.KT       | 408            | 4.58          |
| 23    | Winitrm.BX  | 97             | 1.04          |
| 24    | Yuner.A     | 800            | 8.57          |

B. CLASSIFICATION RESULT

We utilized some standard classifiers for the unbalanced Malimg dataset. The result is shown in Table 3. Random Forest (RF) classifier achieves the highest accuracy of 99.40%, while Nearest Centroid runs fastest with merely 0.11 seconds with an accuracy difference of 1.26% compared to RF in the 10-fold Cross-Validation. Table 8 depicts a confusion matrix that gives the detailed performance of the proposed method using the Random Forest classifier. As can be seen, 22 out of 24 families attain F-scores greater than 90%, 88.1%, and 89.2% of Swizzor.gen!E and Swizzor.gen!I, respectively.

The balanced Malimg dataset of results is shown in Table 4. Even though the number of data is reduced dramatically, we still achieve high accuracy of 98.40% when using the RF classifier. The result shows that our method can extract crucial features of image-based malware. Compared to the previous study, our proposed method reduces by 1% while [38] reduces four times by 4%. The results of the unbalanced Malimg
As shown in Table 7, the Lightweight CNNs of Roseline et al. [29] proposed with merely 0.83M parameters, but the result does not change sharply since the first-time dataset was introduced by Nataraj et al. [12] by 0.31% from 97.18% to 97.49%. That proves that using only a few parameters is not necessarily extracting enough features of the object. On the other hand, utilizing a model with enormous parameters such as ResNet-50 [30] and VGG19 [25] improved the result slightly; however, it requires more computational power. Nevertheless, using a sufficient number of parameters, our lightweight proposed model improves accuracy significantly and saves the computational cost. Moreover, the time to classify each malicious code only takes an average of 0.01s.

Complex architectures such as [25], [30], [32] require high image quality and computational processing capacity. The reason for using complex networks is that the deep layers are expected to extract specific features such as ears and eyes in image processing tasks concerned with humans. On the other hand, the shallow layers focus on overall image features such as edges of the objects. For example, in Fig. 2, many uncomplicated elements can be found by observing the simple grayscale of malware samples. Therefore, we focus on the first layers to extract adequate features with a smaller image size of $64 \times 64$, still ensuring high accuracy.
The Malimg dataset contains many samples processed through obfuscation techniques such as encryption and packing. Among them, malware samples belonging to Adialer.C, Autorun.K, Lolyda.AT, Malex.gen!J, VB.AT, Yuner.A are packed with the same packing process, making them have similar structure and pattern. As a result, analysts often have difficulty distinguishing them. However, our method can process these samples directly without unpacking, with the corresponding accuracy of 100%, 100%, 100%, 99.26%, 99.75%, and 100%, respectively. The experiment indicated that our method was robust against these specific obfuscation attacks.

Moreover, despite achieving high total accuracy of classification, many studies have encountered an obstacle in classifying two family variants: Swizzor.gen!E and Swizzor.gen!I, which are highly similar and difficult to distinguish. The accuracy of both families compared with other authors is shown in Table 6. We achieve the best performance with 87.5% and 87.9% accuracy, respectively.

V. CONCLUSION

Recent studies have developed huge complex neural network models for malware analysis to obtain desirable features. However, they demand more resources than the average system can provide. Therefore, this paper focuses on building simple, lightweight models while still ensuring high classification performance. We propose a feature selection method called A VAE. A VAE consists of a small CNN, variational autoencoder, and an attention mechanism.

Experimental results show that our method could classify malware families efficiently. Our method has achieved the best accuracy of 99.40% with the Random Forest classifier, while Nearest Centroid reaches nearly 99% in under a second. Furthermore, with merely 80 images of each family, our method achieves a high accuracy of 98.40%, which is consistent with the fact that some new families lack data. The total time to convert malicious code into an image (with common malicious code under 1 Mb in size) and classify it merely takes 0.02s. We think our method will be applicable to the existing systems from these results.

Another advantage of our method is that it can distinguish similar malware families with high accuracy even when it is packed. Therefore, our proposed method can help malware analysts reduce the time to classify variants. Furthermore, when the malware family is identified, it is possible to know the typical characteristics, the intended utilization, and the impact of the malware on the target.

In the latent space of VAE, the global features are organized in a more planned way than in AE. However, the importance of the elements has not been considered. In this study, we further emphasize the importance of attention mechanism in selecting and evaluating weights for VAE to help features acquire important features in latent space. At the same time to ensure feature diversity, we combined light-weight CNN to capture lower-range features. Compared to image data generated by malicious code, there are not many complex factors that need a deep CNN network, such as face images or animal images in ImageNet data. The complementary method from the two models helps us acquire rich and different characteristics of the object.

We will build a new malware dataset with recent malicious code for future work. Additionally, we will apply the proposed method to the IDS system to enhance the
capacity for detection and classification of potential dangers in cybersecurity.

In this paper, we have built a model focusing on the issue of classifying malware with simple but effective architecture. We think there is a possibility to apply our method to standard image classification even with the lack of data.

REFERENCES

[1] J. Lee and J. Lee, “A classification system for visualized malware based on multiple autocoder models,” IEEE Access, vol. 9, pp. 144786–144795, 2021.

[2] G. Xiao, J. Li, Y. Chen, and K. Li, “MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks,” J. Parallel Distrib. Comput., vol. 141, pp. 49–58, Jul. 2020.

[3] A. Moser, C. Kruegel, and E. Kirda, “Limits of static analysis for malware detection,” in Proc. 23rd Annu. Comput. Secur. Appl. Conf. (ACSAC), Dec. 2007, pp. 421–430.

[4] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, and W. Aigner, “A survey of visualization of systems for malware analysis,” in Proc. Eurographics Conf. Visualizat. (EuroVis), 2015, pp. 105–125.

[5] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A survey on automated dynamic malware-analysis techniques and tools,” ACM Comput. Surv., vol. 44, no. 2, pp. 1–42, 2008.

[6] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A survey on malware detection using data mining techniques,” ACM Comput. Surv., vol. 50, no. 3, pp. 1–40, May 2018.

[7] L. Liu, B.-S. Wang, B. Yu, and Q.-X. Zhong, “Automatic malware classification and new malware detection using machine learning,” Frontiers Inf. Technol. Electron. Eng., vol. 18, no. 9, pp. 1336–1347, Sep. 2017.

[8] M. Tang and Q. Qian, “Dynamic API call sequence visualisation for malware classification,” IET Inf. Secur., vol. 13, no. 4, pp. 367–377, Jul. 2019.

[9] M. Mimura, “An improved method of detecting macro malware on an imbalanced dataset,” IEEE Access, vol. 8, pp. 204709–204717, 2020.

[10] T. K. Tran and H. Sato, “NLP-based approaches for malware classification from API sequences,” in Proc. 21st Asia Pacific Symp. Intell. Evol. Syst. (IES), Nov. 2017, pp. 101–105.

[11] A. F. Agarap, “Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification,” 2017, arXiv:1801.00318.

[12] L. Nataraj, S. Karthikeyan, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in Proc. 6th Int. Symp. Visualizat. Cyber Secur. (VisSec), 2011, pp. 1–7.

[13] F. Carlo García and F. P. Muga II, “Random forest for malware classification,” 2016, arXiv:1609.07770.

[14] L. Nataraj, S. Karthikeyan, and B. S. Manjunath, “SATTVA: SpArsiTy inspired classification of malware VxRains,” in Proc. 3rd ACM Workshop Inf. Hiding Multimedia Secur., Jun. 2015, pp. 135–140.

[15] G. Conti, E. Dean, M. Sindu, and B. Sangster, “Visual reverse engineering of binary and data files,” in Proc. 5th Int. Workshop Visualizat. Comput. Secur. (VisSec), Jan. 2008, pp. 1–17.

[16] D. Vu, T. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Massacci, and P. H. Phung, “HIT4Mal: Hybrid image transformation for malware classification,” Trans. Emerg. Telecommun. Technol., vol. 31, no. 11, Nov. 2019, Art. no. e3789.

[17] H. Naeem, F. Ullah, M. R. Naeem, S. Khalid, D. Vasan, S. Jabbar, and S. Saeed, “Malware detection in internet of Things based on hybrid image visualization and deep learning model,” Ad Hoc Netw., vol. 105, Aug. 2020, Art. no. 102154.

[18] Y. Ding, X. Zhang, J. Hu, and W. Xu, “Android malware detection method based on bytecode image,” J. Ambient Intell. Humanized Comput., Jun. 2020, [Online]. Available: https://link.springer.com/article/10.1007/s12652-020-02196-4#article-info.

[19] D. Vasan, M. Alazab, S. Wassan, B. Safaee, and Q. Zheng, “Image-based malware classification using ensemble of CNN architectures (IMMEC),” Comput. Secur., vol. 92, May 2020, Art. no. 101748.

[20] A. Vavasi, N. Shazeer, N. Parmar, and J. Uszkoreit, “Attention is all you need,” in Proc. NIPS, 2017, pp. 1–11.
[43] V. S. P. Davuluru, B. N. Narayanan, and E. J. Balster, “Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs,” in Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON). Dayton, OH, USA, Jul. 2019, pp. 273–278.

[44] B. N. Narayanan and V. S. P. Davuluru, “Ensemble malware classification system using deep neural networks,” Electronics, vol. 9, no. 5, p. 721, Apr. 2020.

TUAN VAN DAO was born in Thái Bình, Vietnam, in 1992. He received the B.E. and M.E. degrees from the Department of Computer Science, National Defense Academy, Japan, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree in information security. His research interests include computer vision, artificial intelligence, cyber security, and machine learning.

HIROSHI SATO received the degree in physics from Keio University, Japan, and the master’s and Doctor of Engineering degrees from the Tokyo Institute of Technology, Japan. He is currently an Associate Professor with the Department of Computer Science, National Defense Academy, Japan. Previously, he was a Research Associate at the Department of Mathematics and Information Sciences, Osaka Prefecture University, Japan. His research interests include agent-based simulation, evolutionary computation, and artificial intelligence. He is a member of the Japanese Society for Artificial Intelligence (JSAI), the Society of Instrument and Control Engineers (SICE), and the Institute of Electronics, Information and Communication Engineers (IEICE). He was an Editor of IEICE and SICE.

MASAO KUBO received the Doctor of Engineering degree from Hokkaido University. He is currently an Associate Professor with the Department of Computer Science, National Defense Academy, Japan. His studies on multi-agent system and swarm intelligence.

***